PROYECTO DE SISTEMAS INFORMÁTICOS
CURSO 2010/2011

Shop & Go

Aplicación Móvil Android para Optimización de Rutas en Supermercados

Autores:
Miriam Rioja Bejerano
Jesús Calzada Martínez
Alejandro Rodríguez Martín

Directora
Dra. Victoria López

Jesús Calzada Martínez, Miriam Rioja Bejerano y Alejandro Rodríguez Martín, alumnos matriculados en la asignatura de Sistemas Informáticos, autorizan a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria como el código, la documentación y/o el prototipo desarrollado todo ello realizado durante el curso académico 2010-2011 bajo la dirección de María Victoria López López, profesora del Departamento de Arquitectura de Computadores y Automática de la Facultad de Informática de dicho organismo.

Jesús Calzada Martínez Miriam Rioja Bejerano Alejandro Rodríguez Martín

Este trabajo se lo dedicamos a nuestras familias,
que nos han apoyado en estos años de carrera.
Gracias.

Agradecimientos

Durante todos estos años de carrera se ve muy lejos el final de la misma. Una vez que llega echas la vista atrás y te das cuenta de todas las horas de trabajo y esfuerzo que has invertido. Este proyecto ha sido el final de este aprendizaje, con el cual hemos disfrutado y a la vez peleado desarrollándolo.

Queremos agradecer a nuestras familias y parejas todo el apoyo recibido sobre todo durante este último año de carrera y que han supuesto una gran motivación para no rendirnos en los momentos más duros.

También queremos agradecer a nuestra tutora Victoria López López por su orientación a lo largo del presente trabajo, y a Pablo Pizarro y Diego García sus aportaciones a la interfaz gráfica del proyecto. Por último dar las gracias a la fundación UCM por el material prestado para la realización del trabajo.

Resumen

En este proyecto se ha desarrollado una aplicación para teléfonos móviles ejecutable en el sistema operativo Android que resuelve el problema del camino óptimo en recorridos en un supermercado en el momento de la realización de la compra. La aplicación se completa con otras funcionalidades relacionadas como la confección de listas de la compra, selección y eliminación de artículos, el cálculo (acumulado y total) del coste de la cesta de la compra, la elección de un algoritmo de búsqueda y la delegación de la compra a terceros por medio de envío semiautomático de mensajes de correo electrónico.

La aplicación se desarrolla a partir de una base de datos de artículos situados en un mapa y los precios de los artículos, estos datos son proporcionados por el supermercado en el momento de acceder al centro por tecnología Bluetooth o similar.

Palabras clave: *Aplicaciones móviles, Android, Problemas de rutas, Problemas de caminos mínimos, Supermercados.*

Abstract

This project has developed an executable application for mobile phones Android operating system that solves the problem of optimal path in a supermarket at the time of making the purchase. The application is completed with other related features such as making shopping lists, selection and removal of items into current list, the calculation (accumulated and total) of the cost of the shopping, the choice of a search algorithm and the delegation of purchase from third parties by sending semi-automatic emails.

The application is developed from a database of items located on a map and their prices. The map and other data are provided by the supermarket at the time of accessing the centre via Bluetooth or the Internet.

Keywords: *Mobile Applications, Android, Routing Problems, Shortest Path Problem, Supermarkets.*

Índice general

Capítulo 1
Introducción

En la actualidad la tecnología móvil se encuentra en pleno desarrollo y prueba de ello son los millones de aplicaciones existentes para el teléfono móvil que nos permiten realizar un sinfín de utilidades. La realidad es que el teléfono móvil constituye una herramienta indispensable en nuestra vida y esta representación ha ido cambiando a lo largo de los últimos años. Desde 1983 que apareció el primer móvil hasta la actualidad la tecnología ha evolucionado permitiendo teléfonos más pequeños y con mayores prestaciones. En sus inicios los teléfonos móviles se utilizaban para comunicarse por medio de una llamada de voz o un servicio de mensaje corto (sms) mientras que ahora se utilizan funciones que no hace mucho parecían imposibles, como juegos, reproducción de música, correo electrónico, MMS, agenda electrónica PDA, fotografía digital y video digital, videollamada, navegación por Internet y hasta Televisión digital.

Una aplicación móvil es un programa informático destinado a realizar una función en un dispositivo hardware adecuado. En sus comienzos las aplicaciones móviles estaban destinadas al entretenimiento mientras que en el presente se desarrollan servicios más útiles como callejeros, guías de productos, información actualizada de noticias…. Actualmente las aplicaciones más innovadoras están relacionadas con la *realidad aumentada* ya que ofrece innumerables formas de interacción. La *realidad aumentada* define una visión del mundo real a la que añade información virtual, una aplicación de ejemplo podría ser una capaz de interactuar con los objetos de un museo con la simple acción de sostener el teléfono en su campo de visión.

Desde el punto de vista informático un teléfono móvil es un dispositivo inalámbrico electrónico con una conexión a una red que permite la comunicación con otros dispositivos. Estos dispositivos cuentan con muchas funciones y cada vez es mayor la competencia de distintas empresas por lanzar al mercado más y mejores utilidades.

Debido a su gran penetración en el mercado, las aplicaciones móviles se han convertido en servicios de primer orden. Las principales empresas de telefonía móvil han creado portales y programas en los que poder comprar o descargar todo tipo de aplicaciones. Entre las más importantes están la Itunes Store de Apple [10] y la Android Market de Google [1] pero también hay otras como Ovi Store de Nokia [15] o App Place de Toshiba [3]. Itunes Store que es una tienda online de contenido digital de la empresa Apple, desde 2003 a través de su programa Itunes puedes descargar a tu dispositivo móvil Apple cualquier contenido de entre más de 300.000 aplicaciones de las que un tercio

de ellas son gratuitas. Android Market, una aplicación integrada en los dispositivos móviles Android desarrollada por Google que permite descargar aplicaciones, gratuitas o no, desarrolladas por terceros.

Como realización del proyecto de fin de carrera de la asignatura sistemas informáticos y debido al gran impacto que ha supuesto en el mercado de la telefonía móvil durante los últimos años hemos elegido el sistema operativo Android para desarrollar nuestra aplicación. Android es un producto de Google, en concreto de la *Open Handset Alliance* [14], una alianza formada por aproximadamente 30 organizaciones dispuesta a instaurar una plataforma abierta, integral y gratuita creada específicamente para dispositivos móviles. El hecho de que sea abierto, permite el desarrollo de cualquier aplicación y el uso en el teléfono móvil. Las aplicaciones para Android se diseñan en Java, por lo que nuestra aplicación está implementada en este lenguaje.

Java es un lenguaje de programación orientada a objetos desarrollado por Sun Microsystems [21] a principios de los años 90 y ampliamente utilizado en el entorno académico.

El objetivo principal del proyecto, presentado en esta memoria, es crear una nueva aplicación que nos permita optimizar el recorrido en un supermercado de cualquier persona que posea un móvil con el sistema operativo Android.

Un supermercado es un establecimiento para la venta al por menor de artículos alimenticios y uso doméstico, en el que el cliente se sirve por sí mismo y paga a la salida. Un supermercado está dividido en pasillos, en los cuales cada pasillo posee una gran cantidad de estanterías y en dichas estanterías se encuentran los productos. Los productos se encuentran ordenados en los estantes por secciones: fruta, alimentos congelados, bebidas, etc.

La aplicación que se presenta en esta memoria y a la que nos referimos con el acrónimo Shop&Go está diseñada para facilitar la tarea de realizar la compra en un supermercado de grandes dimensiones. Cada supermercado dispone de una organización distinta en la que dispone sus productos, por lo que muchas veces supone una tarea difícil encontrar un determinado producto sin perder más tiempo de lo debido.

La idea es que el propio supermercado facilite la aplicación Shop&Go para poder ofrecer a sus clientes la posibilidad de realizar la tarea de la compra en un menor tiempo y con mayor facilidad.

Capítulo 2
Estado del arte

1. Problemas de rutas

Los problemas de rutas son una de las partes más importantes de la logística de transporte. Tratan de buscar la optimización de una o varias rutas a realizar por uno o varios vehículos.

Este tipo de problemas pertenece a los problemas de optimización combinatoria, que es una rama de la optimización en matemáticas aplicadas y en ciencias de la computación. Está relacionada con la investigación de operaciones, la teoría de algoritmos y la teoría de la complejidad computacional. Por ello existen múltiples aplicaciones en el campo de la inteligencia artificial, entre otros.

Estos problemas se modelizan habitualmente mediante grafos, y se pueden distinguir en dos clases principales, los *Problemas por Vértices* y los *Problemas por Arcos*.

Un grafo está formado por un conjunto finito de vértices y un conjunto finito de aristas. Un vértice o nodo es un extremo de la conexión formada por alguna arista, se representa mediante un punto. Una arista es la conexión entre dos vértices y se representa mediante una línea que une dos vértices. Además es necesario definir un camino como una lista de vértices unidos por una arista mientras que un ciclo es un camino de longitud no nula que comienza y termina en el mismo vértice. Por ejemplo en la Figura 1 se puede ver un grafo definido por 4 puntos que representan los vértices y 4 líneas que representan las aristas.

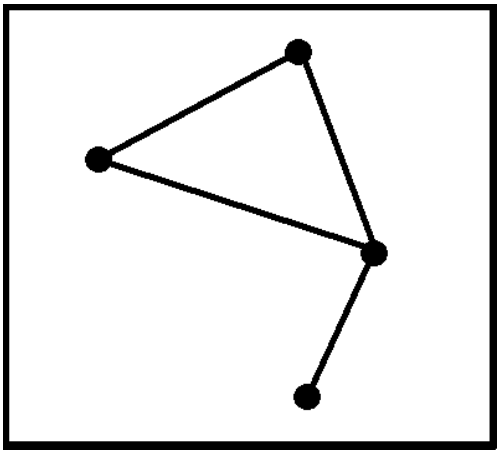

Figura 1. Ejemplo de grafo.

En los Problemas por Vértices el objetivo es recorrer parte o todos los vértices que forman el grafo, pudiendo recorrerlo de manera determinada por el fin del problema de manera que la distancia recorrida en el camino sea lo menor posible. Un ejemplo de este problema es el famoso *Problema del Viajante (TSP, Traveling Salesman Problem)*, el objetivo de este problema en encontrar un camino que empezando y terminando en un vértice concreto y pasando una sola vez por cada uno de sus vértices minimice la distancia recorrida. Para éste y otros muchos problemas se utiliza un grafo en el que cada arista tiene asociada un peso o distancia de tal manera que la distancia de un camino se calcula como la suma de todas las aristas por las que pasa.

En los *Problemas por Arcos* el objetivo es recorrer todos o parte de las aristas de un grafo. El ejemplo para este caso sería el de un cartero que tiene que recorrer todas las calles de una parte de la ciudad (comúnmente llamado *Problema del Cartero Chino*, o *CPP* en sus siglas en inglés).

Estos problemas tienen diferentes niveles de complejidad debido a que pueden tener distintas restricciones, como pueden ser, entre otras, las R*estricciones de Distancia*, en las que los vehículos están limitados por un tiempo o distancia máxima; *Restricciones de Precedencia*, en las que sea necesario visitar unos nodos o aristas antes que otros; o que las rutas no tengan necesariamente que empezar y terminar en el mismo punto. Debido a estas restricciones no es posible generar un algoritmo que encuentre una solución óptima para todos los casos de resolución de rutas.

Se ha trabajado mucho en investigación sobre la resolución óptima de estos problemas explotando la estructura poliédrica del conjunto de soluciones posibles. El ejemplo que más expone esta situación es del problema del viajante, que se ha conseguido resolver en un tiempo razonable para ejemplos con miles de nodos. Sin embargo estos avances conseguidos con el TSP no son fácilmente extrapolables a otros casos de recorridos de grafos, principalmente por las restricciones que hemos comentado anteriormente, diferentes para cada caso.

Actualmente uno de los campos más fructíferos en la resolución de los Problemas Combinatorios (entre los que se encuentran los problemas de rutas) son los *Algoritmos Heurísticos,* que son utilizados precisamente en los casos en los que no existe una solución óptima para sus restricciones. Los algoritmos heurísticos no garantizan la resolución óptima, pero suelen producir soluciones de calidad demostrada. Esto se ha comprobado en ejemplos que se conoce la respuesta exacta, y el coste de la solución con estos algoritmos no suele superar el 1% en muchos casos. No se puede decir que un método heurístico sea superior a otro en general, ya que uno que en unos casos sea mejor a otros, en otros casos distintos puede ser peor.

Los algoritmos heurísticos en la actualidad no se limitan a encontrar una solución posible, ahora los llamados Metaheurísticos exploran el espacio de soluciones en zonas prometedoras y descartan las que son improbables de encontrar una solución mejorando la calidad computacional del proceso. Estas técnicas son, entre otras, los *Algoritmos Genéticos, la Búsqueda Dispersa, Búsqueda Tabú o la Programación de Memoria Adaptativa.*

2. Aplicaciones móviles Android

Como hemos dicho anteriormente una aplicación móvil es un programa informático destinado a realizar una función en un dispositivo informático. En Android Market las aplicaciones se clasifican en los siguientes tipos: compras, comunicación, cómics, deportes, estilo de vida, finanzas, herramientas, multimedia, noticias y meteorología, ocio, productividad, referencia, salud, sociedad, temas, viajes, demostración y biblioteca de software.

Existen en el mercado cerca de 200.000 aplicaciones Android. Por ejemplo aplicaciones que permiten al usuario hacer su propia lista de la compra. Como es el caso de la aplicación Hungry! [8] que permite escribir y guardar varias listas de compra, tener organizado por colores los productos y es capaz de enviarla por correo o sms a otro equipo. La Figura 2 muestra un menú de esta aplicación donde podemos ver una lista formada por algunos alimentos diferenciados por colores.

Figura 2. Hungry!.

Otra aplicación original es ShopSavvy Barcode Scanner [18], es capaz de reconocer un código de barras mediante la cámara incluida en el móvil y proporcionar una lista de precios y comercios en línea. La Figura 3 muestra el escaneo de un código de barras mediante esta aplicación.

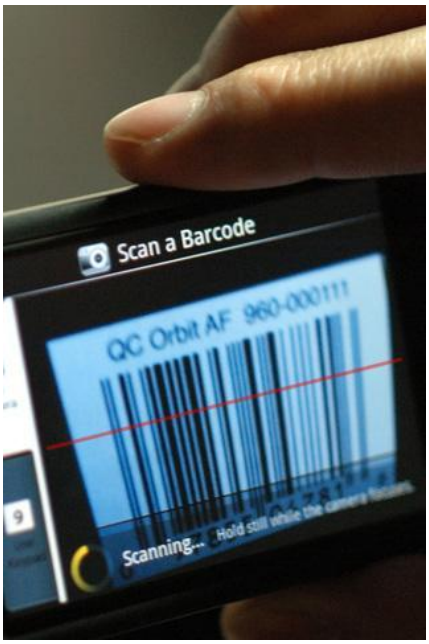

Figura 3. ShopSavvy Barcode Scanner.

Otra aplicación interesante de mencionar es Layar [11] debido a que utiliza la tecnología de *realidad aumentada* a la cual antes nos hemos referido. La aplicación consiste en poder observar el mundo en capas. Estas capas representan un tipo de servicio. Por ejemplo, si seleccionamos la capa de restaurantes cercanos y dirigimos la cámara del móvil a los edificios que tenemos alrededor, en la pantalla del móvil observaremos todos los restaurantes que están en la redonda. También se puede activar una capa para leer los antecedentes y comentarios de los diferentes lugares o símbolos históricos. La Figura 4 muestra la pantalla de un móvil en que se ha seleccionado una capa y la aplicación ha encontrado en el edificio que apunta el móvil un servicio de la categoría seleccionada por la capa.

Figura 4. Layar.

3. Dispositivos móviles

Un dispositivo móvil es un aparato de reducido tamaño, lo suficiente como para ser transportado, que cuenta con capacidades de procesamiento con conexión a una red y una memoria limitada y está diseñado para realizar una o varias funciones.

Durante los años 80, tanto Casio [5] como Hewlett-Packard [9] sacaron al mercado algunas calculadoras programables consideradas como los primeros dispositivos móviles.

La primera PDA desarrollada fue *MessagePad* en 1993 por Apple Computer [4]. Estaban basadas en el procesador *RISC ARM 610* y usaban el sistema operativo *Newton*. Algunas de sus funciones eran transmitir fax y correo electrónico, recibir mensajes, poseía un calendario y agenda electrónica y reconocimiento de escritura. Los siguientes dispositivos móviles en aparecer fueron *Pilot* y *PocketPC*. La Figura 5 muestra el dispositivo *MessagePad*.

Figura 5. MessagePad.

Podemos clasificar los dispositivos móviles en tres grupos.

- Los teléfonos, son los más ligeros y transportables de todos. Su precio también es el más barato aunque un móvil de gama alta puede superar los 400 euros. Sus funciones básicas son recibir y enviar llamadas aunque en la actualidad poseen muchas más aplicaciones.
- Las PDAs también denominadas ordenadores de bolsillo son ordenadores de mano, originalmente diseñados como agenda electrónica aunque actualmente también tienen muchas otras aplicaciones como soporte para leer un libro.
- Por último están las consolas que no sirven sólo para jugar sino que incorporan funciones como reproductor de archivos multimedia, actualmente las dos más importantes son la *Sony PlayStation Portable* [20] (PSP) y la *Nintendo DS* [13].

4. Sistemas operativos móviles

Un sistema operativo móvil es un programa que controla los procesos básicos y necesarios para el funcionamiento de un dispositivo móvil. Cada móvil necesariamente ha de disponer un sistema operativo instalado. Actualmente existen diversos sistemas operativos para móvil, los más importantes los explicamos a continuación.

- Android ha sido desarrollado por Google. Es el sistema operativo más reciente del mercado pero ya consta de 200.000 aplicaciones disponibles para este sistema operativo. Los programas están escritos en Java. Este es el sistema operativo que hemos elegido para hacer nuestra aplicación.

- iPhone OS es un sistema operativo propiedad de Apple desarrollado en 2007 originalmente para el iPhone, siendo después usado en el *iPod Touch* e *iPad*. Los dispositivos que tienen este sistema sólo pueden programarse desde ordenadores con *MacOS* y el lenguaje que suele utilizarse es Objetive C. Actualmente existen mas de 185.000 aplicaciones disponible para iPhone OS.

- Symbian es un sistema operativo para móviles que fue producto de la alianza de varias empresas de telefonía móvil entre las que se encuentran Nokia, Sony Ericsson, Samsung, etc. El desarrollo de aplicaciones para este SO es sencillo ya que permite programar en lenguajes como Java, C++, Visual Basic, Phyton, Perl, Flash Lite… Las plataformas que están basadas en este sistema operativo aportan un *SDK* (Kit de Desarrollo Software) que pueden usar los desarrolladores. Este hecho ha conseguido que existan en la actualidad millones de aplicaciones para móviles Symbian.

- El BlackBerry OS es un sistema operativo móvil desarrollado por Research in Motion [17] para sus dispositivos BlackBerry y está dirigido a su uso profesional. BlackBerry es una línea de teléfonos inteligentes fundamentalmente conocido por el servicio de correo electrónico móvil y el teclado QUERTY. Los lenguajes de programación de este sistema operativo son Java y C++.

- Windows Phone está desarrollado por Microsoft [12] para su uso en Smarphone y otros móviles. Su interfaz de usuario es similar al escritorio de Windows. Anteriormente fue llamado Windows Mobile y a lo largo de su historia ha cambiado de nombre varias veces. Se pueden desarrollar aplicaciones de dos maneras distintas, mediante C++ que usa el API (Interfaz de Programación de Aplicaciones) de Windows Phone o utilizando .NET Compact Framework con C# o Vb.Net.

5. Historia y productos Google

Google Inc. es la empresa propietaria de la marca Google, cuyo principal producto es el motor de búsqueda del mismo nombre. Su principal misión es organizar la información mundial para que resulte práctica y precisa. Google nació en la universidad de Stanford en 1995 como el resultado de la tesis de dos estudiantes. El nombre de Google es un a juego de palabras elaborado a partir del término científico Googol que se utiliza para referirse a un uno seguido de cien ceros, con ello quieren dar a entender la enorme información que representan. Desde entonces Google se convirtió en el buscador más usado de la red superando a AltaVista en 1995. Actualmente tiene una media de doscientos millones de búsquedas al día.

Google es consciente de que el número de móviles triplica el número de ordenadores, por eso no ha dudado en sacar sus principales productos en una versión para el móvil. Así pues podemos encontrar las versiones del buscador de Google, Google Maps y Google Noticias.

Aunque el principal producto de Google es su buscador además ofrece otros muchos productos. Todos estos productos se han ido incorporando en los últimos años y seguro que muchos usuarios no imaginan internet sin estos servicios. Algunos de los más interesantes los explicamos a continuación.

- Gmail es un servicio de correo electrónico desarrollado en 2004 que destaca por su capacidad de almacenamiento, tamaño de archivos adjuntos, chat integrado y su sencilla interfaz.

- Google Maps, es un servidor de aplicaciones de mapas en la Web desarrollado por Google en 2005. Ofrece mapas realizados con satélites del mundo entero, puede calcular la ruta entre distintas ubicaciones y buscar una calle o negocio. Actualmente se ha lanzado Google Maps 5.0

para Android. Como característica más novedosa hay que mencionar que nos permitirá ver mapas en 3D, lo que nos ofrecerá con detalle los edificios y el estado orográfico del territorio.

- YouTube fue creado en 2005 y no es un producto desarrollado por Google pero fue adquirido por él en 2006 por 1650 millones de dólares. En este sitio web los usuarios pueden subir y visualizar videos. Aloja gran cantidad de videos personales, clips de películas, videos musicales… Además los videos pueden recibir votaciones y se pueden realizar comentarios. Actualmente es una de las páginas más visitadas en internet. La Figura 6 muestra un vídeo de YouTube.

Figura 6. Vídeo de YouTube.

- Google Chrome es el navegador web de Google, actualmente es el tercer navegador más utilizado en Internet, al igual que la mayoría de sus servicios está disponible gratuitamente. Su principal objetivo es ofrecer una navegación rápida, estable y segura.

- Google Earth es un programa que permite tener toda la información geográfica y visualizar imágenes en 3D del planeta. Fue desarrollado por la empresa Keyhole Inc. y fue comprado por Google en el año 2004.

También hay que mencionar Google Calendar, Google Documents, Adwords, AdSense, Google Code, Picasa, Picnik, Google Traductor…

6. Programación orientada a objetos y Java

La programación orientada a objetos, abreviado POO, es una propuesta de programación que usa objetos y sus interacciones para poder diseñar aplicaciones y programas informáticos. Esta visión es muy cercana a como expresaríamos las cosas en la vida real.

Uno de los principales lenguajes actualmente basado en la POO es Java. Java surgió en 1991 cuando un grupo de ingenieros de *Sun Microsystems* trataron de diseñar un nuevo lenguaje de programación destinado a electrodomésticos. La reducida potencia de cálculo y memoria de los electrodomésticos llevó a desarrollar un lenguaje sencillo capaz de generar código de tamaño muy reducido. Desarrollaron un lenguaje neutro que no dependía del tipo de electrodoméstico que utilizaban, el cual se ejecutaba sobre una máquina virtual denominada *Java Virtual Machine*.

Como lenguaje de programación para computadores, Java se introdujo a finales de 1995. Desde entonces Java ha experimentado diversos cambios en cada versión.

Java es un lenguaje basado en la programación orientada a objetos además de simple, distribuido, robusto, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico según define la compañía *Sun*.

Las principales reglas de las que está compuesta la POO son clases, atributos, métodos y objetos. Los datos se pueden organizar en clases y las clases se organizan en proyectos o paquetes. Todo programa diseñado con POO está formado por clases, estas clases representan todos los objetos de un tipo particular. Existen distintos tipos de clases dependiendo de la privacidad y seguridad que conste en ella, por ejemplo pueden ser privadas, protegidas, públicas… Además también existen clases abstractas o clases con herencia. Una clase está compuesta por atributos y métodos y puede instanciarse en un objeto. Un objeto es un ejemplar de una clase, se crea mediante un método denominado constructor. Los atributos son las características que se definen para un tipo particular, cuando definimos un atributo tenemos que especificar un nombre y el tipo del atributo. Como tipos se encuentran *int*, *char*, *boolean*... Los métodos son las acciones mediante las cuales un objeto se relaciona con otros objetos, los métodos pueden ser constructores, de acceso (*getter* y *setter*) o creados por el desarrollador. Un método constructor tiene como finalidad inicializar todos los atributos de una clase y devolver un objeto con esos

atributos. Los métodos además pueden ser de tipo *void* o devolver algún tipo, esto significa que si el método no necesita devolver ningún valor deberá poner en la cabecera de su método la palabra reservada *void*. En cambio si el método necesita devolver un valor deberá indicar en la cabecera el tipo del dato que devuelve.

Por ejemplo si pensáramos en una persona, la clase sería persona porque representa un tipo concreto, los atributos nombre, color de pelo, peso y estatura porque son características propias de una persona que pueden variar de una persona a otra, los métodos comer y pensar porque son acciones que puede realizar cualquier objeto de la clase persona y un ejemplo de objeto sería Juan, la Figura 7 ilustra este ejemplo. Otras características fundamentales de la POO son la herencia y el polimorfismo. La herencia es el mecanismo mediante el cual una clase reutiliza el código de otra clase para implementar una jerarquía, es la relación entre una clase más general y otra más específica. En la Figura 7 se representa la herencia por medio de la clase Profesor y Estudiante que heredan de la clase Persona, ya que todos los profesores y estudiantes son además personas con todas sus características y acciones. El polimorfismo se refiere a la posibilidad de que varias clases derivadas de una antecesora puedan utilizar un mismo método de forma diferente.

En el libro *Programación orientada a objetos con Java* [16] se puede encontrar un completo desarrollo del tema de la POO.

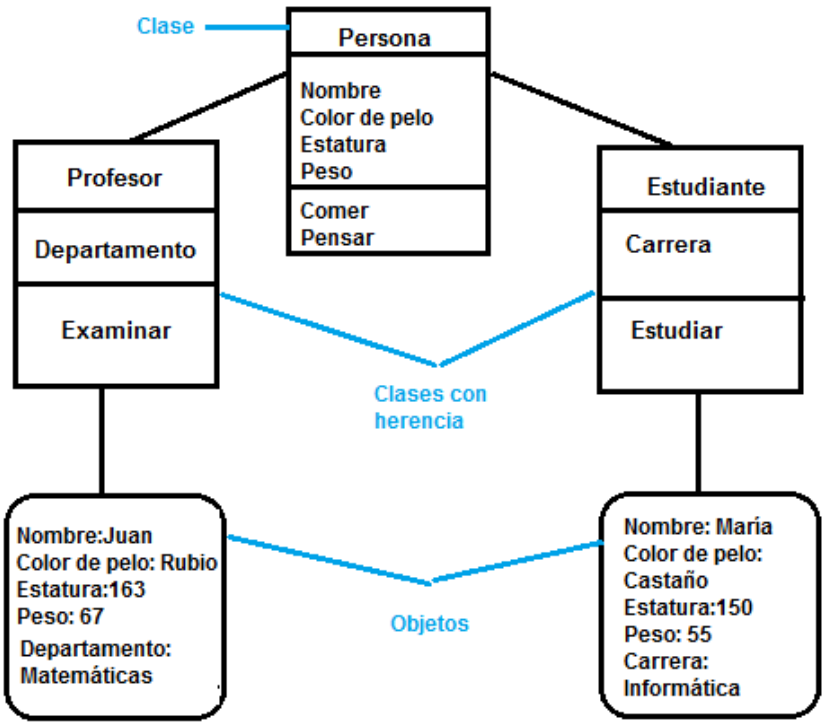

Figura 7. Ejemplo de POO.

Capítulo 3
La plataforma Android

1. Definición y historia de Android

Android es el principal producto de la alianza comercial de 80 empresas del sector de los dispositivos móviles en 2007, llamada *Open Handset Alliance*, sus componentes son fabricantes de hardware, operadores y empresas de software. Algunos miembros de esta importante alianza son Google, Telefónica, Vodafone, Acer y Asus. Los objetivos de esta alianza son acelerar la innovación móvil y ofrecer una experiencia más rica, menos costosa y mejores teléfonos móviles. *HTC Dream* es el primer teléfono que salió al mercado con la tecnología Android exactamente la versión 1.5. y fue lanzado al mercado en 2008.

Android es un entorno software creado para dispositivos móviles. Inicialmente fue desarrollado por *Android Inc*., pero fue comprada por Google en el 2005. Está basado en un sistema operativo Linux que ofrece gran potencia y funcionalidad. Este sistema operativo está programado en lenguaje C mientras que las aplicaciones para Android se diseñan en Java. Una característica de Android es que no distingue entre las aplicaciones incorporadas y las creadas por lo que permite un aprovechamiento máximo de los recursos disponibles en el dispositivo móvil. Con su naturaleza de código abierto la comunidad de desarrolladores proporciona a Android aplicaciones que ayudan a mejorar los aspectos de los que carece.

En este trabajo se ha elegido utilizar el dispositivo móvil HTC Desire para realizar simulaciones y pruebas con datos sintéticos. La elección es casual ya que el dispositivo ha sido proporcionado por la Oficina de Transferencia Tecnológica de la UCM y el único requisito que se ha solicitado sobre el dispositivo es el sistema operativo Android. La empresa Vodafone® ha financiado su coste.

Estudios previos han concluido que Android liderará el mercado de telefonía móvil en los próximos años, esto no es de extrañar ya que en la actualidad tiene una gran aceptación y reputación en el mercado. La Tabla 1 muestra los datos concluidos por los estudios.

Tabla 1. Cuota de mercado.

Worldwide Smartphone Operating System 2011 and 2015 Market Share and 2011-2015 CAGR (listed alphabetically)

Operating System	2011 Market Share	2015 Market Share	2011-2015 CAGR
Android	39.5%	45.4%	23.8%
BlackBerry	14.9%	13.7%	17.1%
iOS	15.7%	15.3%	18.8%
Symbian	20.9%	0.2%	-65.0%
Windows Phone 7/Windows Mobile	5.5%	20.9%	67.1%
Others	3.5%	4.6%	28.0%
Total	100.0%	100.0%	19.6%

Uno de los propósitos de Android consiste en adentrarse en el mercado de servicios basados en ubicación o GPS. Además otro propósito que Android ha podido superar es el de incluir un navegador que optimice la experiencia web, para ello incluye el motor de navegación *WebKit* [23], líder del mercado, que equipara la navegación de escritorio en el móvil.

Los móviles que actualmente incluyen Android son los *Smarphones* o teléfonos inteligentes. Este tipo de teléfono identifica a los móviles que tienen funciones más avanzadas como por ejemplo navegación web, correo electrónico, acceso a las redes sociales…

2. Arquitectura

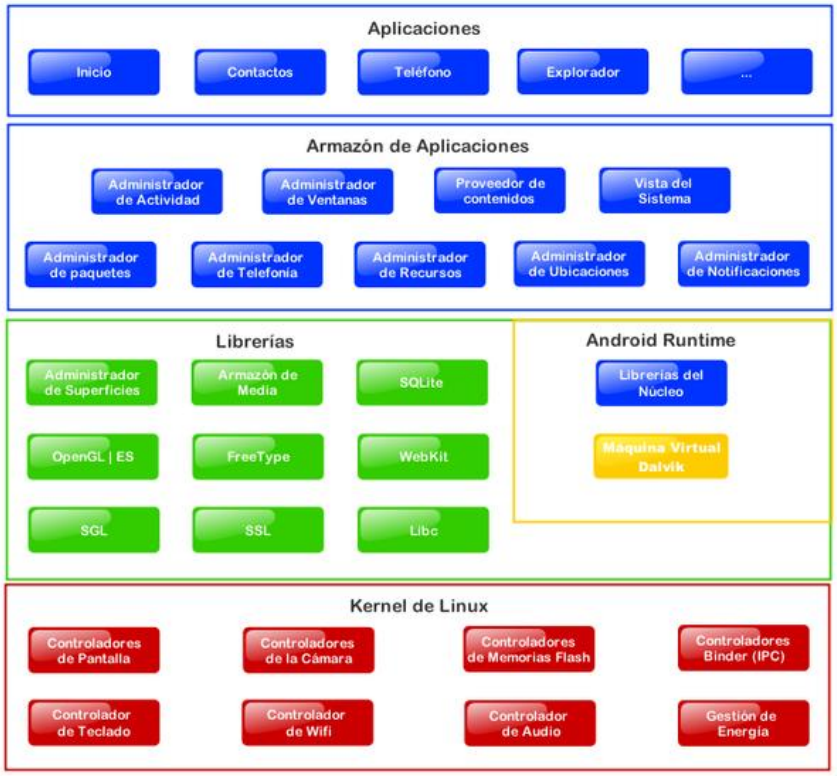

Figura 8. Arquitectura de Android.

La Figura 8 representa el esquema de la arquitectura de Android que pasaremos a explicar a continuación. La capa más profunda está formada por el *Kernel* de *Linux*. Este núcleo es usado para comunicarse con la capa de hardware. A través de esta capa y mediante los drivers necesarios se pueden gestionar procesos, dispositivos, memoria y controladores. La siguiente capa corresponde a *Android Runtime* que está compuesto por las librerías del núcleo y por la máquina virtual *Dalvik*. Las aplicaciones Android se ejecutan en su interior de tal manera que un mismo dispositivo puede soportar varias máquinas virtuales ejecutándose a la vez. Al mismo nivel que la capa Android Runtime se encuentran las librerías de Android. La capa *Application Framework* contiene todas las *API´s* necesarias para implementar las aplicaciones. Por último, la capa de las aplicaciones, donde se encuentran el gestor de inicio, el gestor SMS, el calendario, los contactos, etc. Todas las aplicaciones están escritas en lenguaje Java.

3. El SDK de Android

El *SDK* de Android, acrónimo de *Software Development Kit*, es el conjunto de herramientas de desarrollo de Android. Con este conjunto de herramientas se puede desarrollar un programa que se puede ejecutar en cualquier dispositivo con tecnología Android. El *SDK* se puede descargar gratuitamente en Google en la página

http://code.google.com/android/download.html

Android *SDK* se puede utilizar para las plataformas Windows, Mac Os y Linux.

Para poder realizar este proyecto se ha utilizado el *SDK* junto al entorno de programación Eclipse [6]. Un entorno de desarrollo es un programa informático formado por un conjunto de herramientas de programación. Un *IDE*, *Integrated Development Environment* consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. El software Eclipse es un *IDE* de código abierto multiplataforma para crear Aplicaciones de Cliente Enriquecido. Existen varias versiones de este software, nosotros hemos utilizado para este proyecto la versión *Helios*, lanzada al mercado en 2010 y que actualmente es la última versión disponible.

Se pueden encontrar en el kit de desarrollo *SDK* un conjunto de herramientas que enumeramos en la subsección siguiente.

3.1. Paquetes básicos

Un paquete es un conjunto de funciones agrupadas. Algunos de los paquetes básicos de Android son:

- java.lang: clases básicas del lenguaje Java.
- java.io: funciones de entrada/salida.
- java.net: conexiones de red.
- java.net: clases de red.

- javax.security: clases relacionadas con la seguridad.
- android.opengl: clases OpenGL.
- android.graphics: primitivas gráficas.

Algunos de estos paquetes son fundamentales para el correcto desarrollo de cualquier aplicación, mientras que otros se utilizan en función del tipo de aplicación creada.

3.2. Perspectiva Java

Una perspectiva es un conjunto de ventanas situadas ordenadamente ocupando toda o parte de la pantalla, relacionadas con el funcionamiento de una determinada acción.

El entorno de programación Eclipse posee varias perspectivas, dependiendo del trabajo que se quiera realizar el usuario seleccionará la más apropiada. Las ventanas y herramientas incluidas en la perspectiva Eclipse se denominan vistas. Al desarrollar aplicaciones Android, hay dos perspectivas de Eclipse especialmente relevantes: la *perspectiva Java* y la *perspectiva DDMS*. Para cambiar la perspectiva activa se utiliza el menú *Open perspective* en el *IDE* de Eclipse.

A través de la *perspectiva Java* se desarrollan las aplicaciones Android. Existen diversas vistas para poder elegir la que más se adapte a la funcionalidad que se desea a la hora del desarrollo. Es decir, una vista es una ventana que muestra una determinada información o permite realizar una determinada tarea. Una perspectiva estará formada por un conjunto de vistas distribuidas en la pantalla. Por ejemplo la vista *Package* Explorer permite ver los proyectos de Java en el espacio de trabajo. Otro ejemplo es la vista *Problems* de la perspectiva donde se muestras los posibles problemas de la compilación de la aplicación que se está desarrollando. A partir de esta *perspectiva Java* podemos trabajar con el código fuente de Java y será la perspectiva más utilizada a lo largo del desarrollo de una aplicación Android.

3.3. Perspectiva DDMS

Esta perspectiva nos aporta las funciones del dispositivo mientras está en ejecución. Entre otras cabe destacar las siguientes vistas: La vista *Devices* muestra todos los dispositivos que están en línea y conectados a Eclipse. En la vista *LogCat* podemos encontrar todos los mensajes del dispositivo. La vista *File Explorer* permite administrar archivos del sistema. Por último la vista *Emulator Control* permite probar distintas características de conectividad de redes de voz y de datos, como por ejemplo simular llamadas entrantes o SMS para realizar pruebas. La Figura 9 muestra esta perspectiva en el entorno Eclipse.

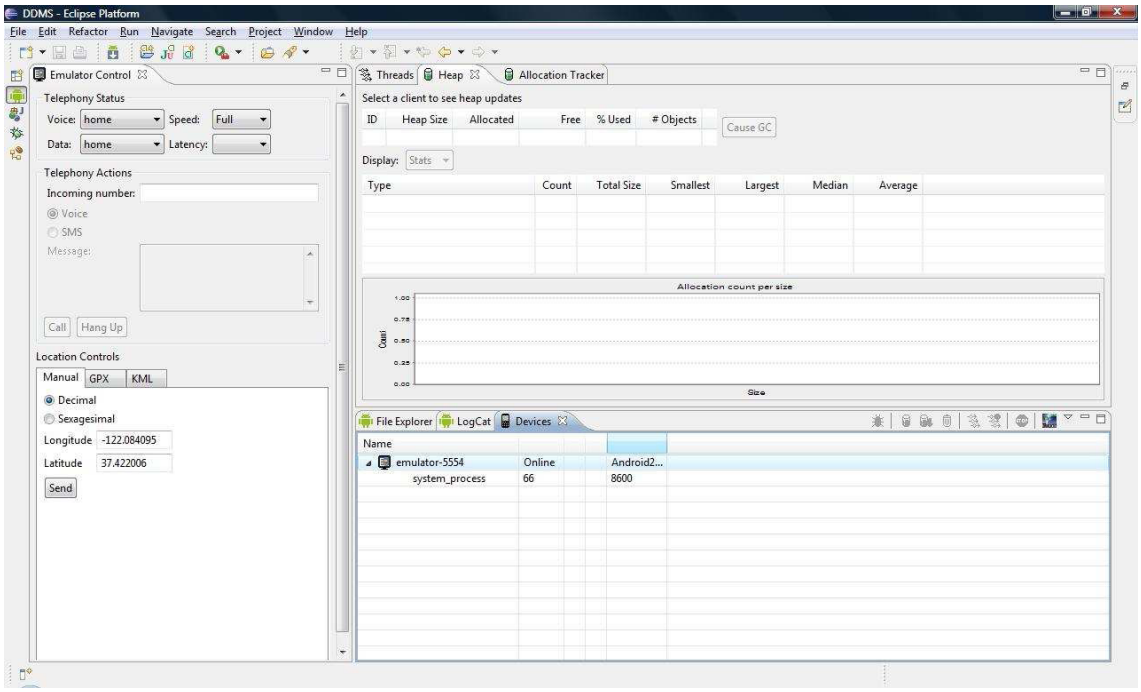

Figura 9. Perspectiva DDMS.

3.4. Emulador

Un emulador permite ejecutar un programa de una determinada plataforma en una diferente para la cual fue creado.

En este caso el emulador Android permite emular y probar aplicaciones Android sin la necesidad de tener instalada la aplicación en un dispositivo móvil. Cualquier emulador es una buena opción para el desarrollador ya que facilita su trabajo notablemente. Además es una opción más rápida y limpia que probar la aplicación sobre el terminal ya que facilita el borrado de archivos y mantenimiento del sistema.

Aunque es la mejor opción para probar una aplicación, tenemos que tener en cuenta las siguientes restricciones: no emula conexiones *USB*, batería o conexión *bluetooth*. En el caso de incorporar una de estas funciones en la aplicación se utilizara el propio dispositivo móvil para realizar el testeo.

Como cada dispositivo móvil tiene sus propias características, el emulador de Android permite cambiar sus diseños para adaptarse a un determinado terminal. Así pues se pueden ajustar, por ejemplo, la resolución de la pantalla o la velocidad de red.

3.5. Debugging

El *SDK* de Android proporciona bastantes de las herramientas que se necesitan para depurar una aplicación. Si se está utilizando Eclipse, ya está incluido un dispositivo llamado *JDWP* que permite depurar una aplicación línea a línea. Los principales componentes que forman parte del proceso de debugging son los siguientes:

- La utilidad *adb* que actúa como un intermediario entre un dispositivo y el sistema de desarrollo. Ofrece diversas funciones de administración de dispositivos, como por ejemplo el movimiento y la sincronización de archivos en el emulador.
- *Dalvik Debug Monitor Server (DDMS)*, es un programa gráfico que se comunica con sus dispositivos a través de adb. *DDMS* puede realizar capturas de pantalla, reunir información de los procesos y emular llamadas entrantes o SMS.
- *Android Virtual Devide*, la aplicación debe ejecutarse en una *AVD* de manera que se pueda depurar.

3.6. Aplicaciones de ejemplo

En el *SDK* podemos encontrar más de veinte ejemplos de aplicaciones, como *Snake* o *Wiktionary*. Estas aplicaciones sirven como guía de desarrolladores poco avanzados e inexpertos que necesitan ayuda a la hora de programar su aplicación.

En el ejemplo de la aplicación *Snake* [19] se implementa el conocido juego de la serpiente y es una ayuda para aquellos que busquen información sobre los *View Layouts* y quieran dibujar en la pantalla.

Wiktionary [24] es una aplicación que muestra en la pantalla principal del teléfono cada día una nueva palabra y su significado. Estas pequeñas aplicaciones de uso diario y específico que se muestran generalmente en la pantalla principal o escritorio del teléfono se denominan *widget*. Si el desarrollador estudia este ejemplo encontrará información sobre cómo realizar un *widget*.

4. Versiones

Desde sus inicios Android ha ido evolucionando y aptándose a las necesidades de la tecnología actual, para ello este sistema operativo ha lanzado nuevas versiones mejorando las anteriores y proporcionando nuevas habilidades. Todas las versiones tienen en común que el nombre que las distingue forma parte de un tipo de postre.

- **Android 1.0 y 1.1:** Android sacó su primera versión en septiembre de 2008 llamada Android 1.0. Seis meses después apareció la segunda versión Android 1.1, esta versión incluía algunos cambios estéticos además de soporte para búsquedas por voz y otras mejoras.
- **Android 1.5:** A mediados de mayo de 2009, Google lanza la versión 1.5 llamada *CupCake* que incluía nuevas características y funcionalidades como grabación de video, sistema de teclado personalizado en pantalla o soporte para Bluetooth. Actualmente esta versión todavía se encuentra en uso en móviles como *HTC Hero*.

- **Android 1.6:** La siguiente versión apareció en septiembre de 2009 bajo el nombre de *Donut* con mejoras en las búsquedas, nueva pantalla para controlar la batería y una experiencia mejorada con Android Market entre otros muchos beneficios. Actualmente es una de las versiones más populares.

- **Android 2.0 y 2.1:** A continuación se lanzo la versión 2.0 *Eclair* en noviembre de 2009. Como mejoras cabe destacar el rediseño de la interfaz del navegador, mayor velocidad hardware y soporte para nuevos tamaños y resoluciones de pantalla. En enero de 2010 se lanzó la versión Android 2.1 con mejoras como reconocimiento de voz y galería 3D además de nuevas posibilidades para interactuar con el navegador asociado a ciertos gestos como *drag and drop* (arrastrar y soltar).

- **Android 2.2:** El 20 de mayo de 2010 apareció *Froyo*, la versión 2.2 de Android, introduciendo la optimización general del sistema Android, la memoria y el rendimiento.

- **Android 2.3:** En diciembre de 2010 aparece la versión *Gingerbread* 2.3 que incluye como progreso una nueva actualización del diseño de la interfaz de usuario, nuevos formatos multimedia y nuevas formas de comunicación. En la Figura 10 podemos encontrar los datos recolectados de las versiones utilizadas por los usuarios durante dos semanas terminando el 4 de enero 2011.

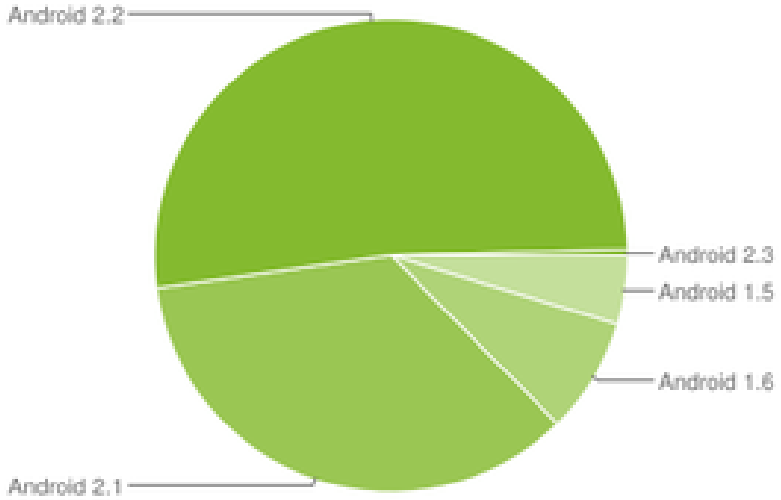

Figura 10. Datos versiones Android. [22]

- **Android 3.0 y 3.1:** La última versión *Honeycomb* dispone de un mejor soporte para *tablets*, un nuevo escritorio en 3D y mejoras en el sistema multitarea y en el navegador web. A fecha de abril de 2011 tan sólo posee un 2% de cuota de mercado.

5. Desarrollo de Android

5.1. Esquema de una aplicación

Todos los proyectos en Android tienen un esquema común organizado en las siguientes carpetas:

- Carpeta *src*: en esta carpeta se encuentra en código de la aplicación.
- Carpeta *gen*: en esta carpeta se encuentra el archivo *R.java*, que contiene los identificadores de cada elemento de la interfaz de usuario definido en la aplicación, este archivo se genera automáticamente.
- Carpeta *res*: donde se sitúan los recursos de la aplicación. Está subdividida en:
    - res/drawable: almacén de imágenes e iconos utilizados.
    - res/layout: ficheros XML que definen layout.
    - res/values: ficheros para traducción de String, colors…
    - res/xml: ficheros XML genéricos.
    - AndroidManifest.xml: archivo que contiene información sobre la aplicación.

5.2. Componentes de una aplicación

Una aplicación Android puede estar formada por uno o varios componentes principales. Además estos componentes deberán ser declarados en el archivo *XML* dispuesto para ello denominado *AndroidManifest.xml*. Los cuatro bloques o componentes principales son los siguientes:

- **Activity**

Un *Activity* es un interfaz de usuario, es decir una pantalla visible donde se refleja una determinada actividad. Una aplicación Android puede tener más de uno de estos elementos. *Activity* representa los elementos de la interfaz de usuario con una *View* o con un *Layout*. *Activity* es el elemento de una aplicación más utilizado.

Un ejemplo de una aplicación que contenga varios *Activity* puede ser una agenda. En una primera pantalla se muestran todos los contactos disponibles, en una segunda pantalla se muestra un determinado contacto con toda la información de éste.

Para poder cambiar de una pantalla a otra se utiliza el elemento *Intent*. Un *Intent* se define como una acción necesaria. Mediante el lanzamiento de un *Intent* una aplicación puede moverse a través de distintos *Activity*. De esta manera los *Activity* podrán estar activos o no. Los *Activity* que no están activos se guardan en una pila, ya que posteriormente pueden volver a ser usados. Una aplicación se puede mover libremente a través de varios *Activity* de tal manera que sólo habrá uno ejecutándose simultáneamente.

- **Service**

Un *Service* se utiliza cuando se requiere una acción prolongada de fondo y no necesita de interfaz de usuario. Un ejemplo de este componente sería la reproducción de música mientras el usuario realiza otras acciones.

- **Broadcast Receiver**

Broadcast Receiver se utiliza para recibir y responder a un evento global cuando éste se produzca, como por ejemplo una llamada de teléfono o un mensaje de texto entrante. Este componente tampoco dispone de interfaz de usuario.

- **Content Provider**

Content Provider se utiliza para gestionar datos tanto de ficheros, bases de datos u otro formato, además de revelarlos a otras aplicaciones. Cualquier clase que implemente *Content Provider* contendrá un conjunto de métodos para que la aplicación pueda acceder a los datos que maneja.

- **AndroidManifest.xml**

Android Manifest es un archivo que genera automáticamente una aplicación de Android. Contiene la configuración de los cuatro elementos anteriores: *Activity*, *Service*, *Broadcast Teceiver* y *Content Provider* para poder ejecutar una aplicación.

6. Ciclo de vida de una aplicación Android

En Android cada aplicación se ejecuta en su propio proceso. Un proceso es un conjunto de actividades organizadas que se realizan con un fin determinado. Android es capaz de evaluar el sistema y determinar en todo momento qué aplicaciones son ejecutadas por el sistema y qué importancia tienen. Así pues, Android determina el tiempo de vida de cada proceso.

Para determinar qué procesos no son ejecutados, Android ordena los procesos que se están ejecutando en orden de importancia. Podemos distinguir los siguientes tipos de procesos ordenados por importancia:

- *Foreground process* (proceso de primer plano): es un proceso que contiene un *Activity* en el primer plano de la pantalla y que el usuario está usando. Esto significa que en un sistema Android sólo podremos encontrar unos pocos de este tipo de procesos corriendo al mismo tiempo. Son los procesos más importantes y que sólo serán eliminados por el sistema en última circunstancia.

- *Visible process* (proceso visible): es un proceso que contiene un Activity, pero no en primer plano. Esto es debido a que el *Activity* ha ejecutado su método que le permite estar en pausa, de tal manera que el proceso está corriendo pero no está siendo usado de manera directa.

- *Service process* (proceso de servicio): es un proceso que contiene un *Service*. Estos procesos no son visibles al usuario pero ejecutan tareas importantes como puede ser la reproducción de música. Es sistema trata de mantener este tipo de procesos a menos que la situación de ocupación de memoria sea grave.

- *Background process* (proceso de fondo): es un proceso que contiene un *Activity* que no es visible por el usuario, esto es debido a que el *Activity* ha ejecutado su método de parar. Estos procesos pueden ser eliminados del sistema de tal manera que no se interrumpa ninguna actividad visible para el usuario. Generalmente hay muchos de estos procesos corriendo a la vez en un sistema por lo que todos estos procesos se guardan en una estructura de tal manera que el último eliminado será el último al que el usuario tuvo acceso.
- *Empty process* (proceso vacío): es un proceso que no contiene a ningún componente activo. Son procesos que han sido ejecutados anteriormente y que se han guardado una copia en el sistema para optimizar la carga de estos si se vuelve a ejecutar la aplicación. Estos son los procesos menos importantes para el sistema Android y son los que eliminará antes.

7. Primera aplicación, Hello World

A continuación estudiaremos a modo de ejemplo de proyecto de Android el famoso *Hello World* que muestra por pantalla dicho mensaje.

7.1 Creación de un nuevo proyecto

Una vez abierto el programa Eclipse, se deberán ejecutar los siguientes pasos:

1. Para crear el proyecto seleccionar la opción *New* del menú principal *File*. En el submenú se deberá seleccionar *Other*. La Figura 11 muestra este paso.

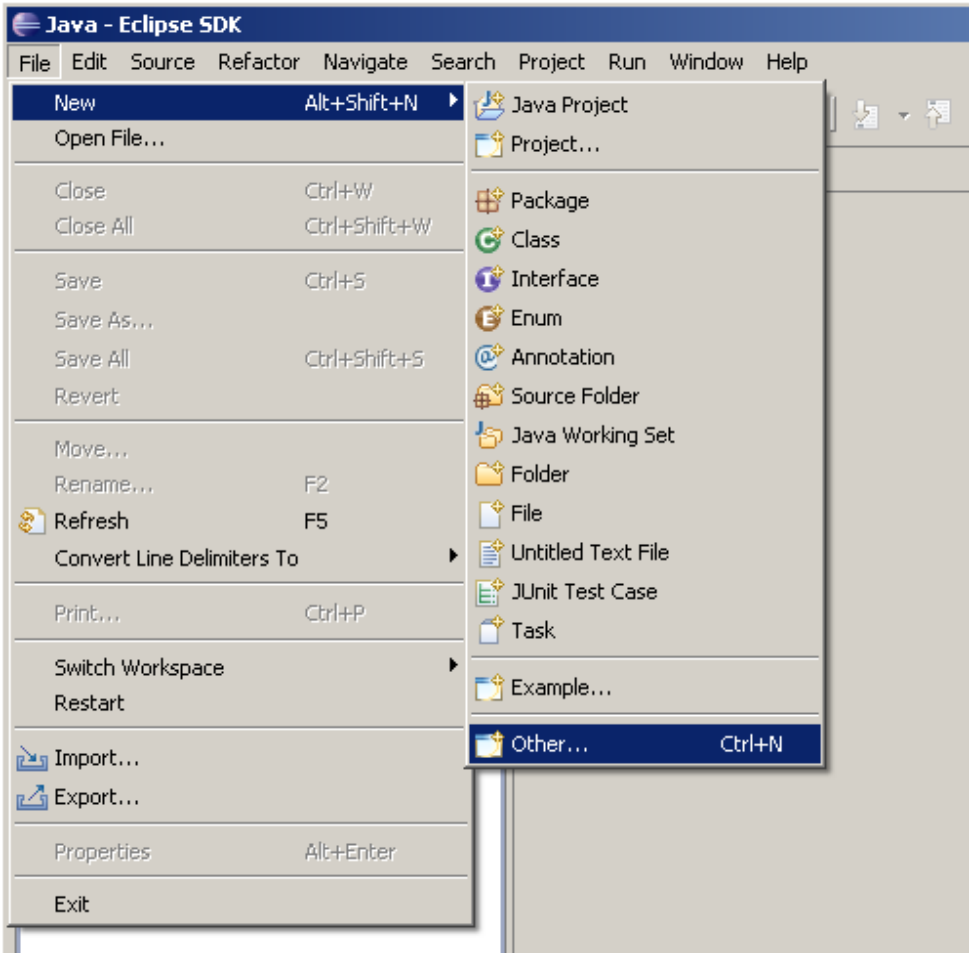

Figura 11. Creación de un proyecto.

2. Aparecerá una nueva ventana en la que se debe seleccionar la opción Android y elegir *Android Project* y pulsar el botón *Next*. La Figura 12 muestra cómo seleccionar dicha opción de la lista.

Figura 12. Selección de proyecto Android.

3. En este paso se deberán configurar las propiedades del proyecto de tal manera que queden como en la Figura 13.

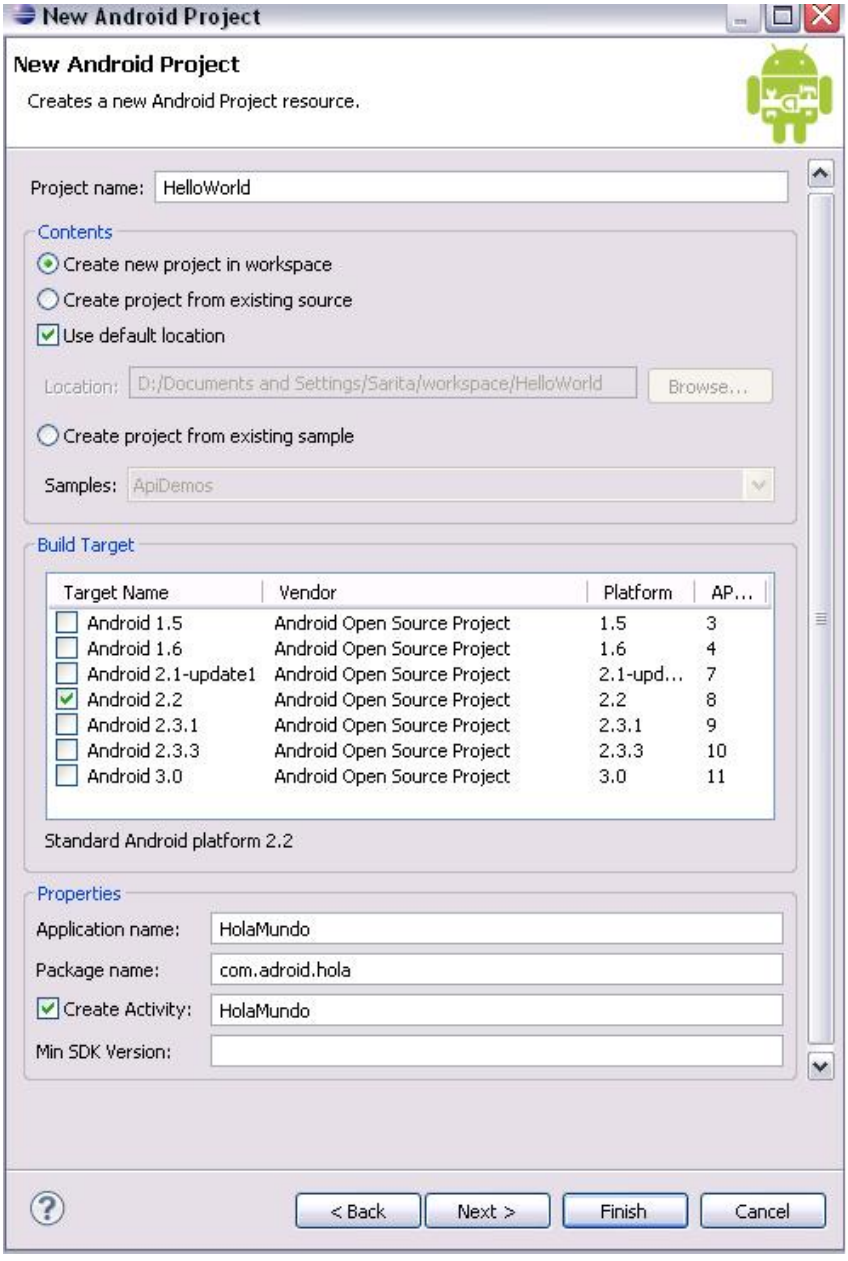

Figura 13. Configuración del proyecto.

- Project name: es el nombre del proyecto. En la práctica, será el nombre que reciba la carpeta donde se guardará todo lo relativo al presente proyecto. En este caso, “HelloWorld”.

- Application name: el nombre de la aplicación que se va a desarrollar. Constituye el nombre visible para el usuario del dispositivo móvil. En el ejemplo, “HolaMundo”.

- Package name: el nombre del paquete bajo el cual será desarrollado todo el código. En este caso “com.android.hola”.

- Activity name: es el nombre de la clase Activity que será creada de forma automática. Esta clase Activity simplemente es una clase ejecutable, capaz de realizar alguna tarea, y es imprescindible en la mayoría de las aplicaciones para Android. Por ejemplo, póngase el nombre “HolaMundo”.

4. Tras configurar estos parámetros deberá pulsar el botón *Finish*. Se puede observar en el nuevo proyecto creado la vista del paquete de explorador. Si despliega el proyecto podrá ver el contenido del mismo. Además a la derecha podrá ver el archivo abierto, en este caso el único *Activity* creado *HolaMundo.java*. En la Figura 14 se puede ver la distribución de la ventana del programa.

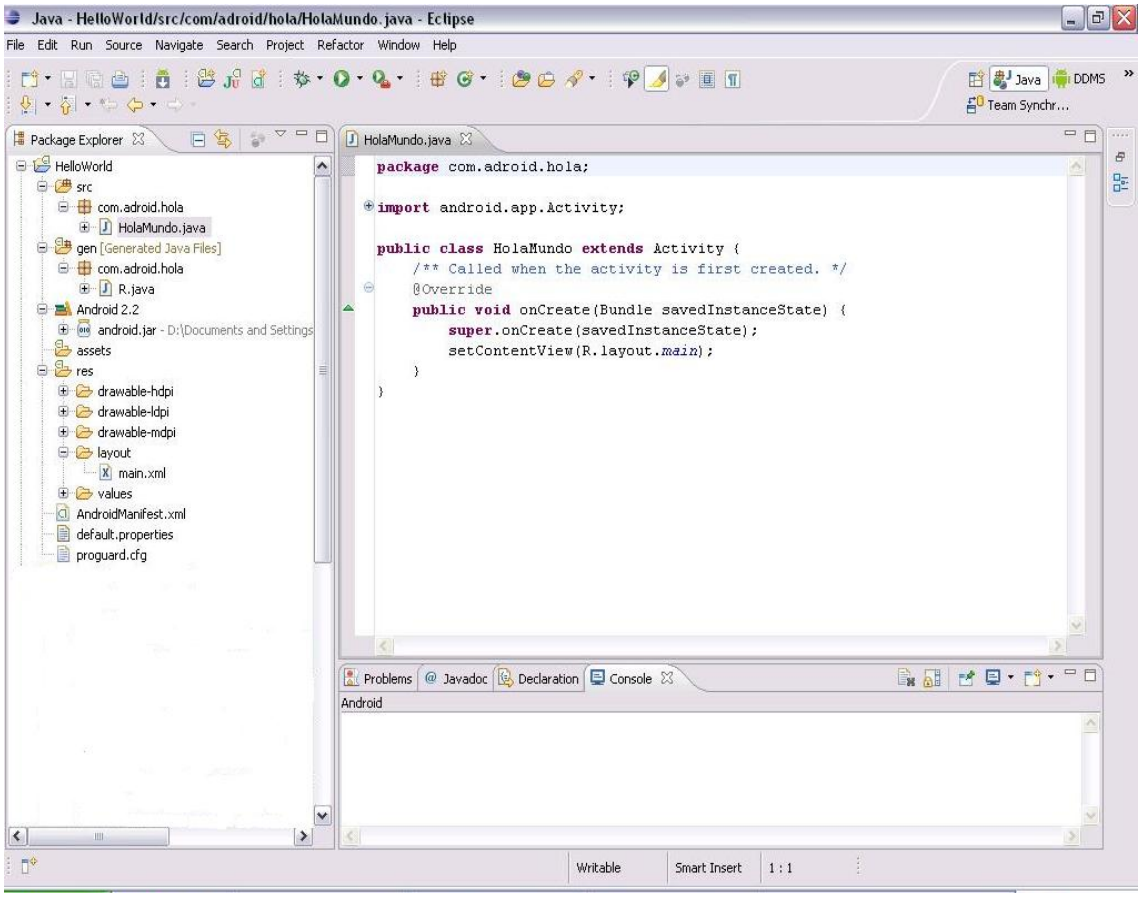

Figura 14. Vista del *Package Explorer* y la ventana de desarrollo.

7.2. Desarrollo de la aplicación

Para continuar con esta primera aplicación de ejemplo, debemos buscar en el explorador la carpeta denominada *src* donde se encuentran los dos ficheros principales de nuestra aplicación: “HolaMundo.java” y “R.java”.

En el fichero “HolaMundo.java” podemos encontrar el siguiente código:

```
package com.android.hola;

import android.app.Activity;
import android.os.Bundle;

public class HolaMundo extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Para poder seguir el ejemplo se debe cambiar el código anterior de manera que resulte el siguiente código.

```
package com.android.hola;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HolaMundo extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hola Mundo");
        setContentView(tv);
    }
}
```

Vamos a analizar cada nueva línea añadida al programa:

```
TextView tv = new TextView(this);
```

Se crea una etiqueta de texto. Al constructor se le pasa como parámetro el contexto de Android. Utilizamos el contexto para tener acceso a datos de carga como base de datos.

```
tv.setText("Hola Mundo");
```

Le asignamos a la etiqueta que se ha creado el valor que queremos en este caso “Hola Mundo” llamando al método setText.

```
setContentView(tv);
```

Mediante este método se asigna la vista a la pantalla.

7.3. Simular la aplicación

Para poder simular la aplicación creada debemos crear una configuración específica. Para simular el proyecto debemos seleccionar la opción *Run Configurations* del menú *Run As* tal y como muestra la Figura 15.

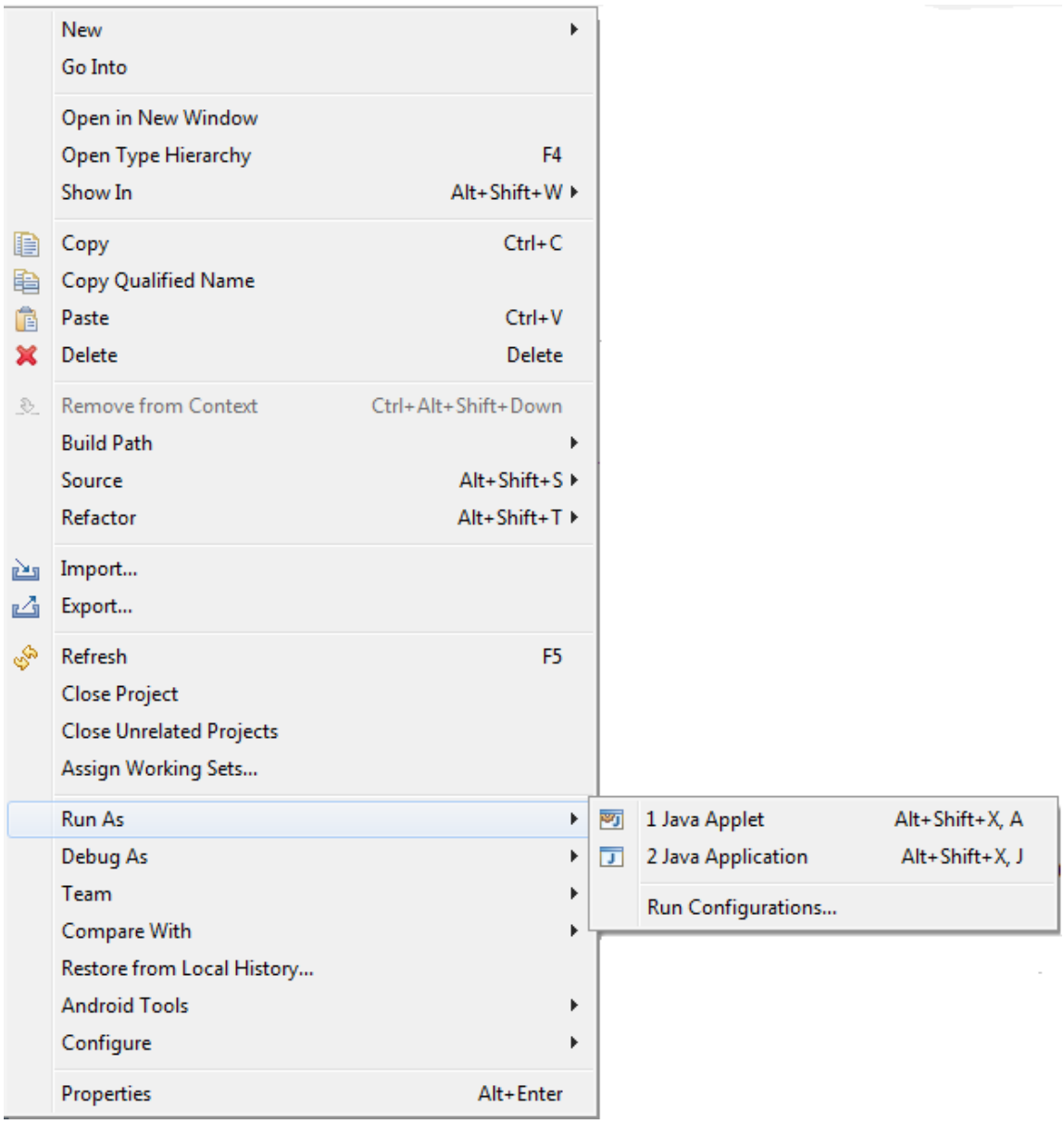

Figura 15. Menú configuración.

En la nueva ventana aparecen tres pestañas que debemos configurar tal y como muestra la Figura 16.

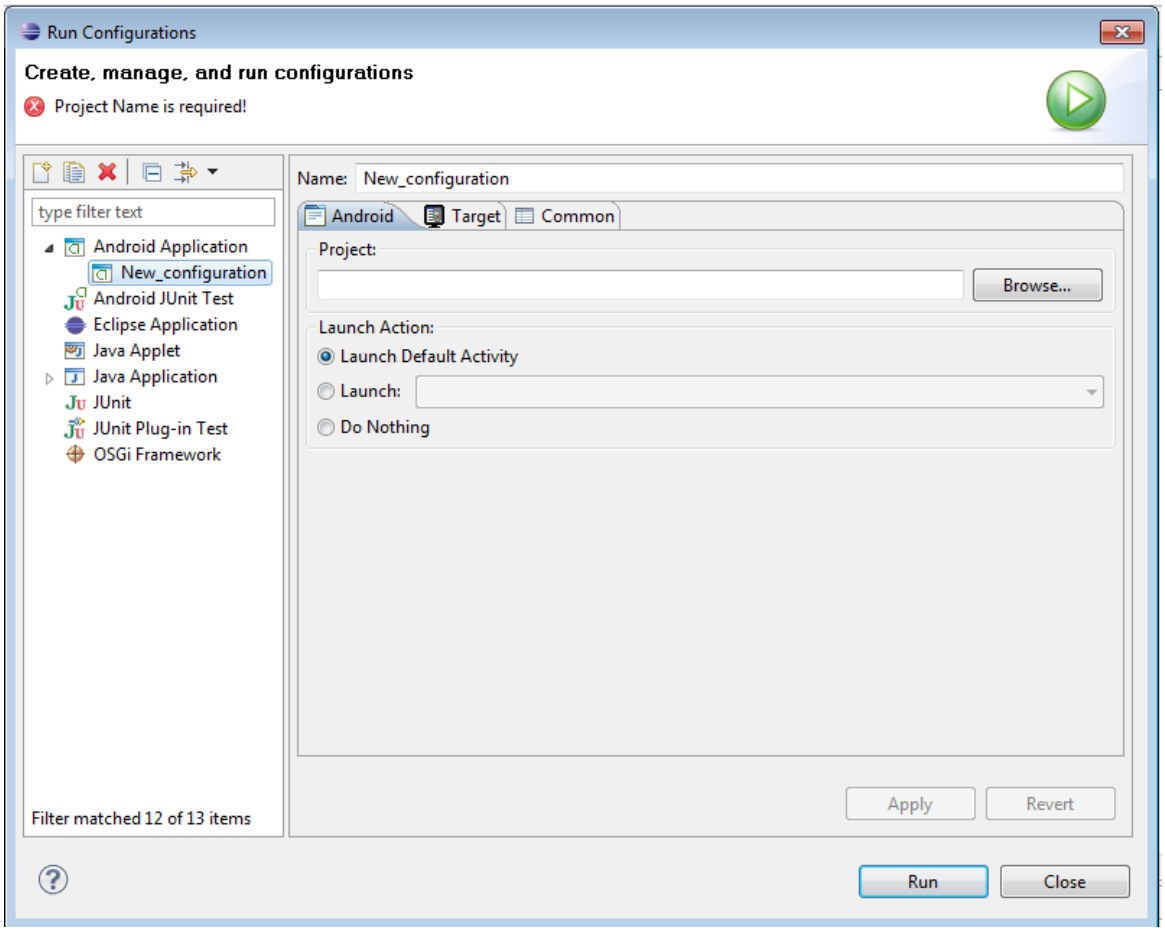

Figura 16. Configuración.

Lo primero es asignar un nombre a esta configuración para poder distinguirla de otra. En la primera pestaña Android se selecciona el proyecto a ejecutar en el botón *Browse*. Aparece una nueva ventana donde se muestran todos los proyectos Android disponibles en nuestro *workspace* y donde se debe seleccionar el proyecto con el que se va a trabajar. Adicionalmente se selecciona el primer Activity que se inicia mediante el campo *Launch,* es decir el Activity que inicia la aplicación. En la segunda pestaña *Target* se selecciona el *skin*, que incluye el diseño de la pantalla, velocidad de red y la latencia. En la tercera pestaña *Common* se puede añadir la configuración al menú de favoritos de Eclipse y seleccionar *Run* y/o *Debug*.

Capítulo 4
La aplicación Shop & Go

1. Especificación de requisitos

1.1. Requisitos funcionales

Los requisitos funcionales se refieren a las acciones que realiza la aplicación y a su funcionamiento, dando a conocer las posibilidades que ofrece al usuario y lo que puede hacer con ella.

En concreto, la aplicación que hemos denominado Shop&Go, ofrece las siguientes funcionalidades al usuario:

- Búsqueda de un producto en el supermercado: El usuario puede elegir un producto de dos maneras distintas, bien buscándolo entre los disponibles en una lista, bien introduciendo su nombre en el campo asignado a tal efecto. A continuación la aplicación muestra un mapa en el que estará señalada la ubicación de dicho producto.

- Creación y edición de listas: Los usuarios pueden crear listas, identificándolas con un nombre, y con la posibilidad de añadir una descripción. Cuando se crea una lista se añaden los productos y con la posibilidad de eliminarlos o añadir nuevos elementos posteriormente. Estos productos proceden de una base de datos que no puede ser modificada por parte del usuario. Las listas creadas serán persistentes, es decir, una vez creadas se guardarán para posteriores consultas en futuras ejecuciones de la aplicación.

- Borrado de listas: En cualquier momento el usuario puede borrar una lista existente.

- Conocer el precio: Una vez que el usuario haya elegido los productos para su lista se mostrará el coste que tendrá la adquisición de los productos.

- Envío de una lista por correo electrónico: Una vez que el usuario ha creado una lista puede enviarla por correo electrónico delegando a otro dispositivo la ejecución del recogido y proceso de compra.

- Elección entre dos algoritmos: A partir de una lista creada, el usuario puede escoger entre dos algoritmos cuya ejecución proporciona un recorrido solución. Al ser dos algoritmos diferentes, los recorridos son, por lo general, diferentes. El problema del circuito mínimo completo de un grafo (problema del viajante, TSP) no posee solución y debe aproximarse mediante algoritmos heurísticos. En esta aplicación se ofrecen dos heurísticas diferentes, con resultados diferentes, pero que pueden visualmente ser seleccionadas por el usuario en función de la percepción de éste. De esta forma, el usuario puede visualizar la información sobre el coste del recorrido manejando también sus preferencias.

- Visualización del recorrido en mapa: Una vez que el usuario ha creado una lista tiene la posibilidad de visualizar un mapa en el que se muestra el recorrido mediante el cual puede adquirir todos los productos de la lista. El usuario podrá ir pulsando en los productos una vez los haya cogido y estos serán eliminados de la lista de forma que se ejecuta el proceso con la interacción del usuario..

- Conocer la distancia recorrida: El mapa proporciona la información adicional sobre la distancia que se va a recorrer.

- Visualización de recorrido en texto: Otra opción de visualización, aparte del mapa, es ver el recorrido en formato texto que indicará el camino a seguir mediante instrucciones sencillas como avanzar o girar a derecha o izquierda.

- Recorrido por voz: Para facilitar el uso de la aplicación a personas con problemas visuales, Shop&Go proporciona la posibilidad de escuchar el recorrido a realizar por los altavoces del móvil.

1.2. Requisitos no funcionales

Los requisitos no funcionales son los criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos. Esto es, requisitos que no tienen que ver con la funcionalidad de la aplicación. En nuestro caso, los objetivos principales a cumplir son:

- Escalabilidad: El sistema será ampliable fácilmente, sin ser necesario recompilar la aplicación. Para poder añadir más productos o un mapa más grande bastará con modificar las bases de datos.
- Rendimiento: Es importante que no haya tiempos de carga largos para evitar la desesperación del usuario y el desinterés consecuente. Además permite aumentar la duración de la batería de los dispositivos.
- Extensibilidad: La aplicación podrá ser ampliada, con nuevas funcionalidades como la descarga de mapas de nuevos supermercados o la actualización de los precios de los productos.
- Capacidad de almacenamiento: Al tratarse de un dispositivo móvil hay que tratar de reducir al máximo el tamaño de la aplicación.
- Facilidad de uso: La aplicación será intuitiva, y se mostrarán las opciones claramente en pantalla, utilizando nombres que hagan fácil la asociación de un botón con la acción que realiza.

2. Diseño

Una vez que tenemos todas las nociones básicas de programar con Android incluso hemos realizado un pequeño ejemplo y hemos configurado adecuadamente el entorno de trabajo pasamos a explicar detalladamente la aplicación que hemos realizado para este proyecto.

2.1. Algoritmo ordenado

Para realizar este algoritmo hemos supuesto un plano de un supermercado ficticio pero cuyas características presentan similitudes es escenarios reales concretamente hemos contado con la ayuda de un plano proporcionado por El Corte Inglés [7] a partir del cual hemos diseñado nuestro mapa.

El plano está formado por 6 bloques, en el cual cada bloque está compuesto por estanterías. Además existen 3 pasillos horizontales y 4 pasillos verticales que separan los 6 bloques de manera que se puedan recorrer todas las estanterías que rodean los bloques. En la Figura 17 podemos ver la distribución de los pasillos y las estanterías.

Las estanterías pueden tener conexión directa por medio de un pasillo o pueden no estar conectadas. Representamos en el esquema los pasillos mediante nodos o puntos rojos con un número y los bloques mediante rectángulos. El punto con el número 0 representa siempre en inicio del supermercado, mientras que el punto con el número 18 representa las cajas.

Para diseñar este algoritmo hemos tenido en cuenta que uno de los recorridos más cortos que puede realizar una persona corresponde al siguiente esquema: dónde el comprador empieza en el nodo número 0 y avanza por los nodos de manera ordenada, es decir intentará hacer un recorrido que avance por los nodos 0-1-2-3… Los nodos están dispuestos en el mapa como muestra la Figura 17, podemos ver en esta figura que los productos más lejanos de la caja están atribuidos a números más pequeños y los productos más cercanos a la caja están atribuidos a los números más grandes. De tal manera que el algoritmo que proponemos, al seguir el orden de dichos números, intenta seleccionar primero los alimentos más lejanos de la caja y dejar para el final los más cercanos a la caja, esto influye positivamente en la distancia recorrida ya que al dejar para el final los más cercanos a las cajas nos aseguramos de no pasar cerca de ellas sin tener la compra completamente realizada. Es decir, este esquema está diseñado para asegurar que el comprador, en caso de querer buscar varios alimentos, deje para los últimos los que estén más cercanos a las cajas. Este algoritmo está pensado y diseñado por los miembros del proyecto. Tras varios intentos fallidos de encontrar un algoritmo útil y eficiente, se optó por la implementación de una variante del conocido *Algoritmo de Dijkstra* [2], para la búsqueda de caminos mínimos en un grafo, desde un nodo concreto al resto de nodos. Este algoritmo realiza la búsqueda mediante

un árbol de expansión cuyas ramas representan los caminos mínimos desde el nodo raíz hasta cada uno de los nodos objetivo. Aunque el objeto de nuestro estudio es un recorrido por un subgrafo de nodos, el conocimiento previo del árbol de caminos mínimos, así como su posible ejecución en cada iteración del problema mayor, ha posibilitado el desarrollo de un algoritmo heurístico tan eficiente como otros que en la literatura resuelven problemas semejantes.

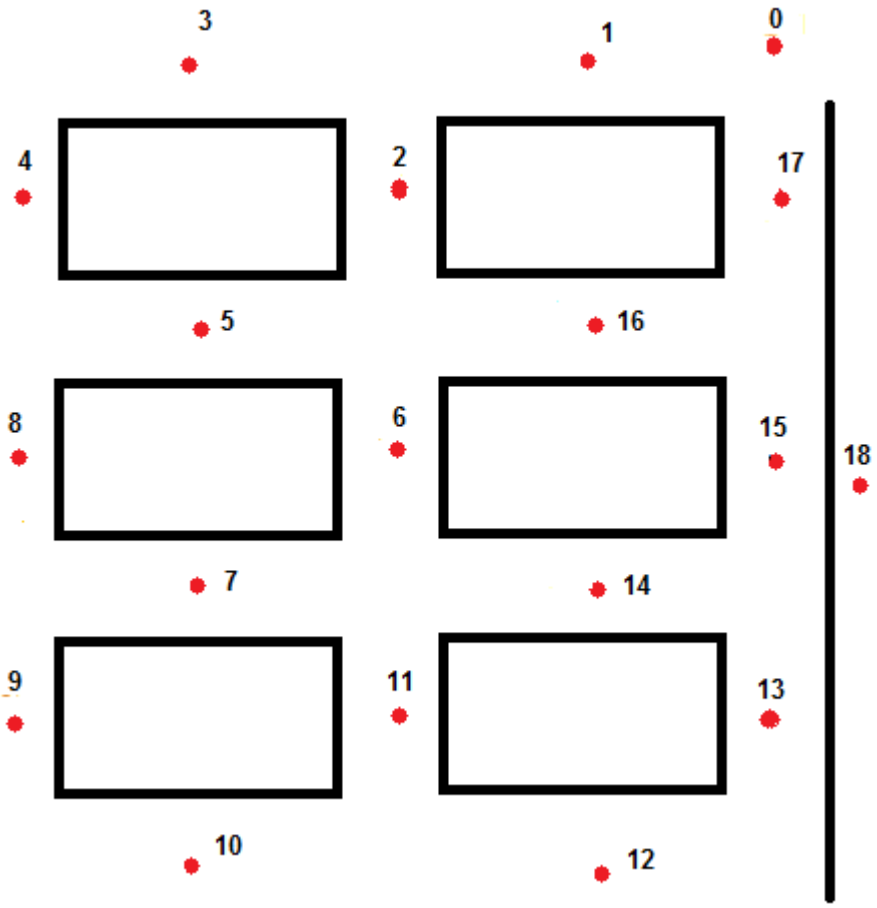

Figura 17. Esquema de supermercado.

La Figura 18 muestra la localización de los productos en el plano que utilizamos para nuestro proyecto. Dicha distribución de los alimentos es totalmente ficticia pero se podría adaptar perfectamente a cualquier distribución real de cualquier supermercado cuyo plano sea facilitado. El texto escrito en cada nodo encima de un rectángulo representa el producto que está situado en ese pasillo y en esa estantería. Los puntos que tienen conexiones son aquellos que están en pasillos contiguos.

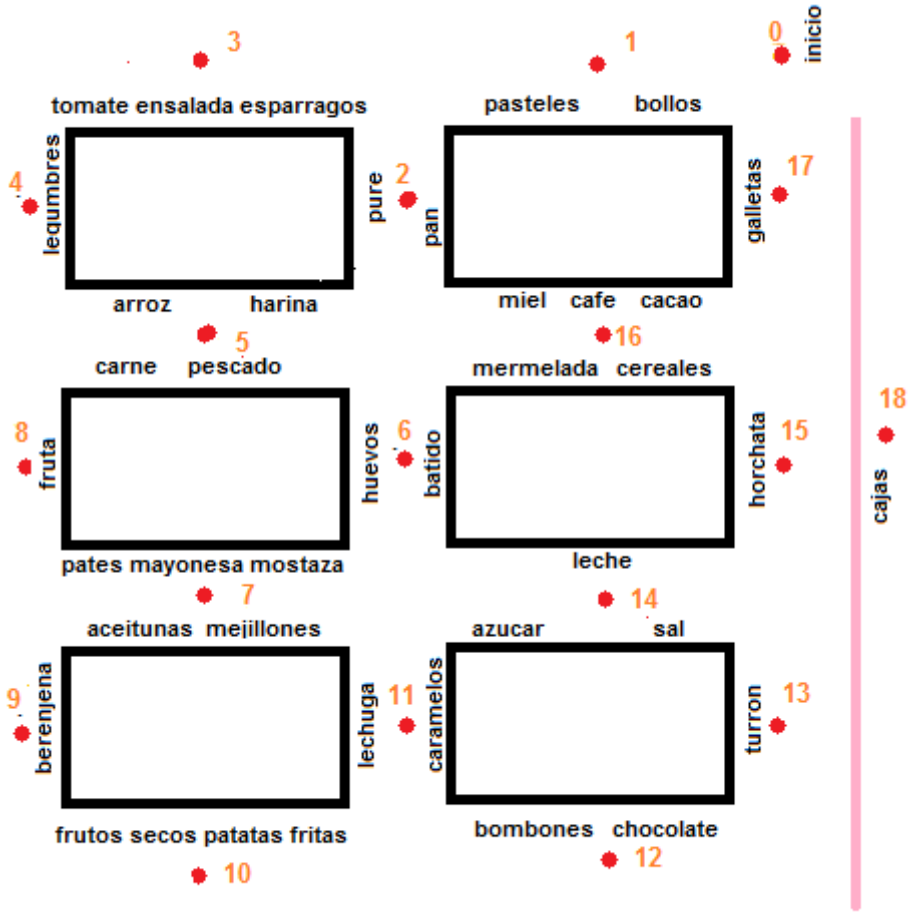

Figura 18. Distribución de los productos.

Para llevar a cabo este algoritmo analizamos primero las estructuras necesarias:

- Una estructura representada en el algoritmo mediante un *ArrayList* denominado *Solución*. Un *ArrayList* en Java se define como una estructura que permite contener y ordenar objetos, posee un tamaño dinámico, es decir, crece a medida que se insertan elementos. Esta estructura denominada *Solución* se inicializa con el nodo 0 ya que el inicio está representado con ese nodo. Durante el algoritmo se irá guardando aquí los nodos intermedios que forman la solución final, y por lo tanto al terminar el algoritmo en esta estructura se encontrara el camino propuesto siendo una sucesión de nodos por los que el comprador debe de pasar. Otra estructura que se necesita es *L*, se usa otro *ArrayList* para representarlo. En *L* se insertan los nodos en los que se encuentra un producto de la lista.

- Además se necesitan varias estructuras que almacenen datos para su posterior consulta. Estas estructuras son las *bases de datos*. Una *base de datos* es un conjunto de datos almacenados para su posterior consulta cuyo acceso es rápido y estructurado. Una de las *bases de datos* que se necesitan es la denominada *DistanciasDbAdapter*, en ella se guardan las distancias entre cada dos pasillos conectados que forma el esquema. Es decir guardamos por ejemplo la distancia entre 0 y 1, 1 y 2, 1 y 3, y así sucesivamente. Además se necesita otra *base de datos* para almacenar los alimentos que se encuentran en las estanterías de cada pasillo, esta *base de datos* se ha denominado *ProductosDbAdapater*. Así por ejemplo en el nodo 12 se guardan los alimentos bombones y chocolate. En el caso de realizar el proyecto sobre otro mapa distinto se tendrá que actualizar todas las *bases de datos*.

- Por último se necesita otra estructura que se calcula cada vez que se ejecuta la aplicación por medio del *algoritmo de Dijkstra*. Por medio de este algoritmo se guarda en un *array* doble denominado *matrizDestinos* los datos obtenidos por él. Un *array* se define como una estructura capaz de almacenar objetos del mismo tipo con un tamaño fijo de longitud. Un *array doble* en programación hace referencia a una *matriz* donde se disponen los elementos en filas y columnas. En cada posición que representa la *matriz matrizDestinos* se guarda el siguiente nodo por

el que tiene que pasar para ir de un nodo origen a un nodo destino. Para el ejemplo nos calcularía lo siguiente (abreviado para 4 nodos):

	1	2	3	4
1	1	1	2	3
2	2	2	2	3
3	3	3	3	3
4	3	3	4	4

Figura 19. Ejemplo de matriz.

La primera fila vertical representa los nodos origen y la primera fila horizontal representa los nodos destino. Las demás filas representan el siguiente nodo que tiene que recorrer para ir desde el nodo origen al destino. Por ejemplo para ir del nodo 2 al nodo 4, se mira en la tabla la fila del 2 y la columna del 4, hay un 3. Eso significa que el primer nodo por el que tiene que ir en su recorrido es el 3. Pero aún no se conoce si es el final. Por lo que se busca en la tabla la fila 3 y el nodo 4, hay un 3, esto significa que hemos llegado al fin y el camino resultante sería 2, 3, 4.

Una vez explicadas todas las estructuras de datos involucradas en el algoritmo se explican a continuación los pasos seguidos en el algoritmo utilizado.

El *algoritmo de Dijkstra*, también denominado algoritmo de caminos mínimos es un *algoritmo* para la obtención del camino más corto dado un vértice origen al resto de vértices en un *grafo* con pesos en cada arista. Su nombre proviene de Edsger Dijkstra quien lo formulo por primera vez en 1959.

- Primero se inicializa el conjunto Solución con el nodo de origen inicio.

- Segundo se hallan los nodos en los que se encuentra algún producto de la lista mediante una pregunta a la base de datos de un producto y éste nos devolverá su número de nodo. Una vez que se tienen todos los nodos sin repeticiones se ordenan de menor a mayor, esto significa que seguirán el camino propuesto en la Figura 20. A continuación se meten estos nodos en el conjunto *L* incluido el nodo caja.

- El siguiente paso que se realiza es un bucle en el que mientras el conjunto *L* no sea vacio se repetirá: se coge el primer nodo de *L* y el último nodo visitado de *Solución*. Se halla el camino más corto entre esos dos nodos mediante la *matriz matrizDestinos*. Se meten los nodos por los que se pasa en *Solución* (y también se acumula la distancia calculada a partir de la matriz de distancias en una variable). Después se borra el nodo que se acaba de utilizar de *L*.

- Cuando *L* esté vacío significa que se ha recorrido todos los nodos por los cuales se debía pasar para recoger el producto de la estantería de la lista de la compra. En *Solución* se obtiene la solución final del algoritmo compuesta por todos los nodos desde el inicio por los que se debe pasar hasta el nodo final de caja.

Para comprender mejor este diseño se realiza a continuación un ejemplo. La lista de la compra está compuesta por tomate, fruta y chocolate. La persona que realice la compra y según el algoritmo propuesto en esta sección deberá realizar el siguiente recorrido según muestra la Figura 20.

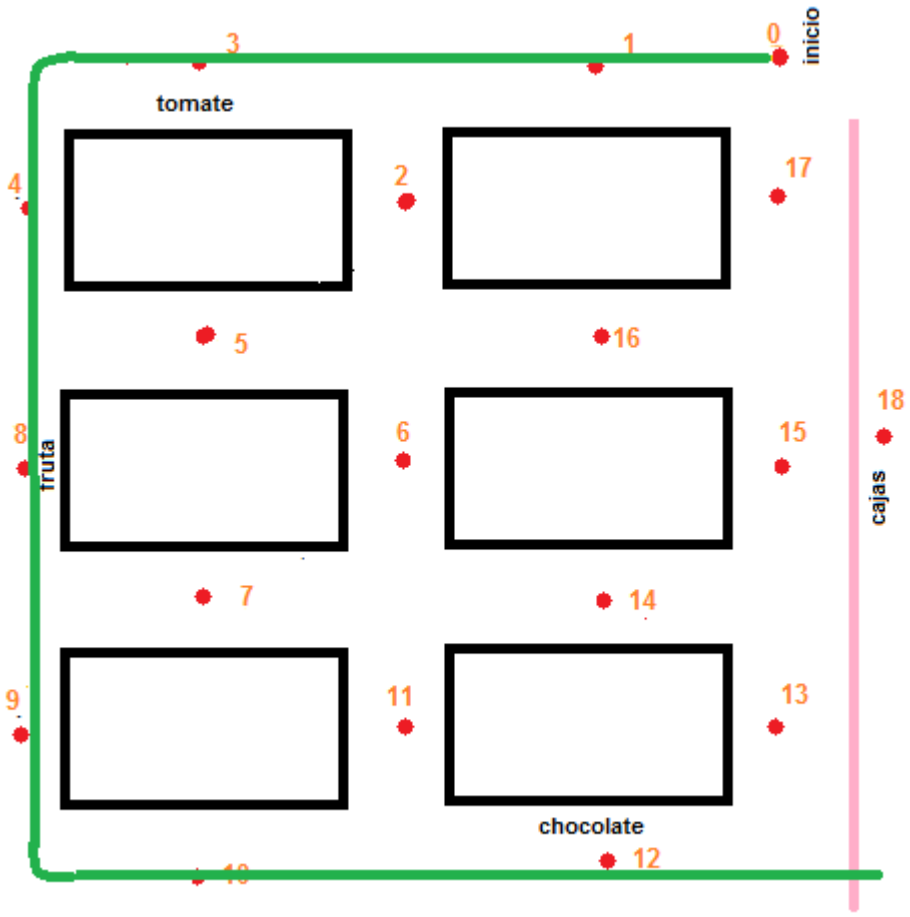

Figura 20. Recorrido de ejemplo.

Inicialmente:

Solución = { 0 }

Buscamos mediante consultas los nodos dónde encontramos los alimentos de nuestra lista, en la Tabla 2 se representa una estructura

Tabla 2. Lista de ejemplo.

Tomate	3
Fruta	8
Chocolate	12

Una vez que se tienen los nodos sin repeticiones y ordenados se añaden el nodo de caja y se inserta en L, obteniendo:

L = {3, 8, 12, 18}

Mientras que L no sea vacío se realizarán los siguientes pasos.

Se seleccionan el último nodo visitado, en este caso el nodo 0 y el primer nodo de L, en este caso el 3. Se calcula el siguiente nodo a visitar para ir desde el nodo 0 al nodo 3 y se inserta en Solución. De tal manera que resulta:

Solución = {0, 1, 3}

Se suprime el nodo 3 de L.

Se selecciona el último nodo visitado, en este caso el nodo 3 y el primer nodo de L, con lo cual se calcula el camino entre el nodo 3 y el nodo 8 y el resultado se añade a Solución quedando:

Solución = {0, 1, 3, 4, 8}

Y así sucesivamente hasta agotar los nodos de la estructura L.

Como solución a este problema se obtiene el camino formado por la siguiente sucesión de nodos:

0, 1, 3, 4, 8, 9, 10, 12, 18

Como se puede comprobar el resultado es el mismo que muestra la Figura 20.

2.2. Algoritmo cercano

Como alternativa al algoritmo anteriormente planteado se ha desarrollado el algoritmo cercano que se explica a continuación, basado tal y como su nombre refiere a la cercanía de los nodos.

Mediante este algoritmo se busca ir al nodo del mapa donde se encuentre el nodo que contiene a un producto de la lista que sea más cercano del punto en el que se encuentra el comprador.

Las estructuras que necesitamos son las mismas que en el algoritmo ordenado. Los pasos a seguir serían los siguientes:

1. Primero se inicializa el conjunto *Solución* con el nodo de origen inicio.
2. Segundo se confecciona la lista de nodos por los que se encuentra un producto de la lista de la compra y se insertan sin repeticiones a la estructura *L*.
3. Hasta que la lista *L* de nodos sea vacía se repiten los siguientes pasos: desde el nodo actual se calcula la distancia a todos los nodos de la estructura *L* y se elige entre ellos el nodo que posea la distancia más pequeña. Se apunta dicho nodo en la estructura Solución y se borra de la estructura *L*. Así se repite sucesivamente para todos los nodos. Una vez que se ha calculado todo el recorrido se debe calcular el camino hasta la caja, añadiendo los nodos necesarios a la solución.

Para una mejor comprensión se explica el siguiente ejemplo: si la lista de la compra contiene los siguientes alimentos: café y berenjena se realiza un recorrido que visita primero el nodo donde se encuentra el café y después la berenjena, aunque éste no sea el algoritmo más óptimo.

Tabla 3. Lista de ejemplo.

Café	16
Berenjena	9

La Tabla 3 representa la lista de ejemplo mediante un producto y el nodo al que corresponde dicho producto.

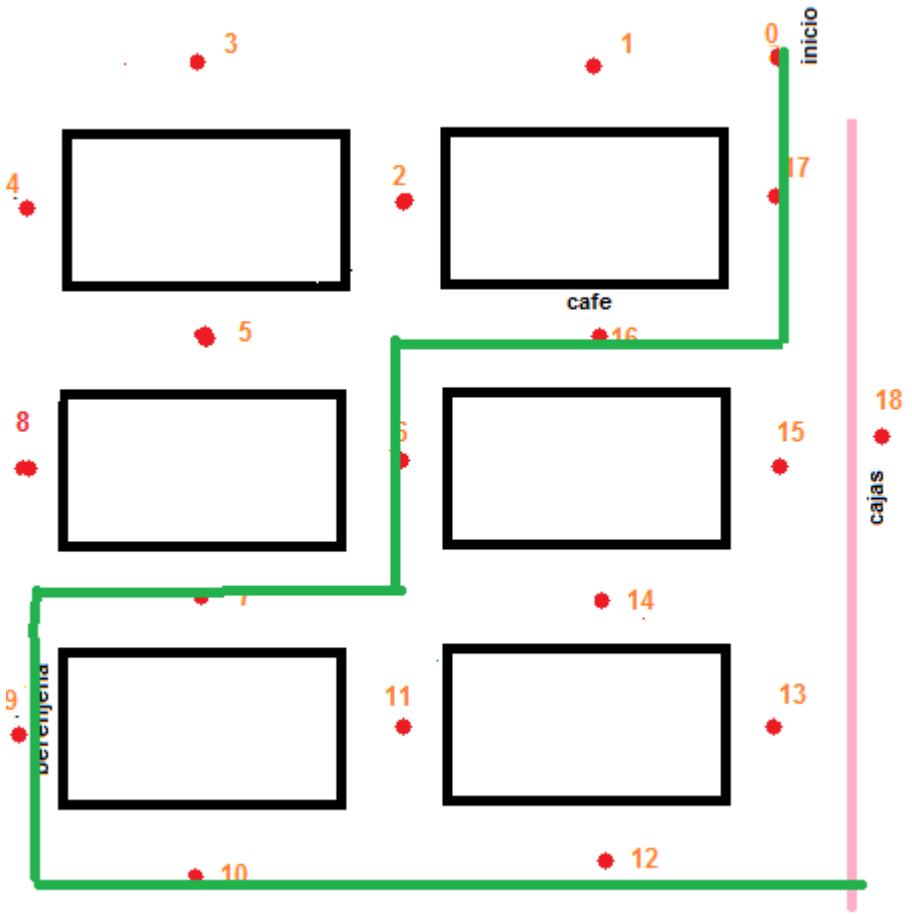

Figura 21. Ejemplo más cercano.

Se inicializa la variable *Solución*:

Solución = {0}

Se calcula la estructura *L* para esta lista en particular y se obtiene:

L = {16,9}

Hasta que *L* no sea vacía se repetirán los siguientes pasos. Se coge el nodo actual en este caso 0 y se calcula la distancia a todos los nodos de *L* tal y como muestra la Figura 22.

Nodo origen	Nodo destino	Distancia
0	16	5
0	9	13,5

Figura 22. Cálculo de la distancia.

Como la distancia del nodo 0 al nodo 16 es menor se apunta este nodo 16 como siguiente nodo a recorrer y se borra de *L*.

Se calcula el recorrido desde 0 hasta 16 mediante la *matriz* de destinos calculado con el *algoritmo de Dijkstra* y así se obtiene como solución:

Solución = {0, 17, 16}

Como *L* no es vacía se vuelve a calcular la distancia a todos los nodos de L desde el último nodo visitado, en este caso 16 tal y como muestra la Tabla 4.

Tabla 4. Cálculo de la distancia.

Nodo origen	Nodo destino	Distancia
16	9	10

Como la distancia del nodo 16 al nodo 9 es la menor se apunta este nodo 9 como siguiente nodo destino y se borra de *L*.

Se calcula el recorrido desde el último nodo visitado, en este caso el 9 hasta las cajas y se añade a la estructura Solución, finalizando así la obtención del recorrido mediante este método y obteniendo como resultado el siguiente, que es el mismo que representa la Figura 21:

Solución = {0, 17, 16, 6, 7, 9, 10, 12, 18}

2.3. Comparativa entre los algoritmos

A continuación se evalúan la eficiencia de los dos algoritmos anteriormente explicados mediante el estudio de varios ejemplos.

- Ejemplo 1

En la lista de la compra se tienen los siguientes productos: ensalada, arroz, fruta, bombones y miel.

Con el algoritmo ordenado se obtiene el siguiente recorrido tal y como muestra la Figura 23:

0, 1, 3, 2, 5, 8, 9, 10, 12, 11, 6, 16, 18

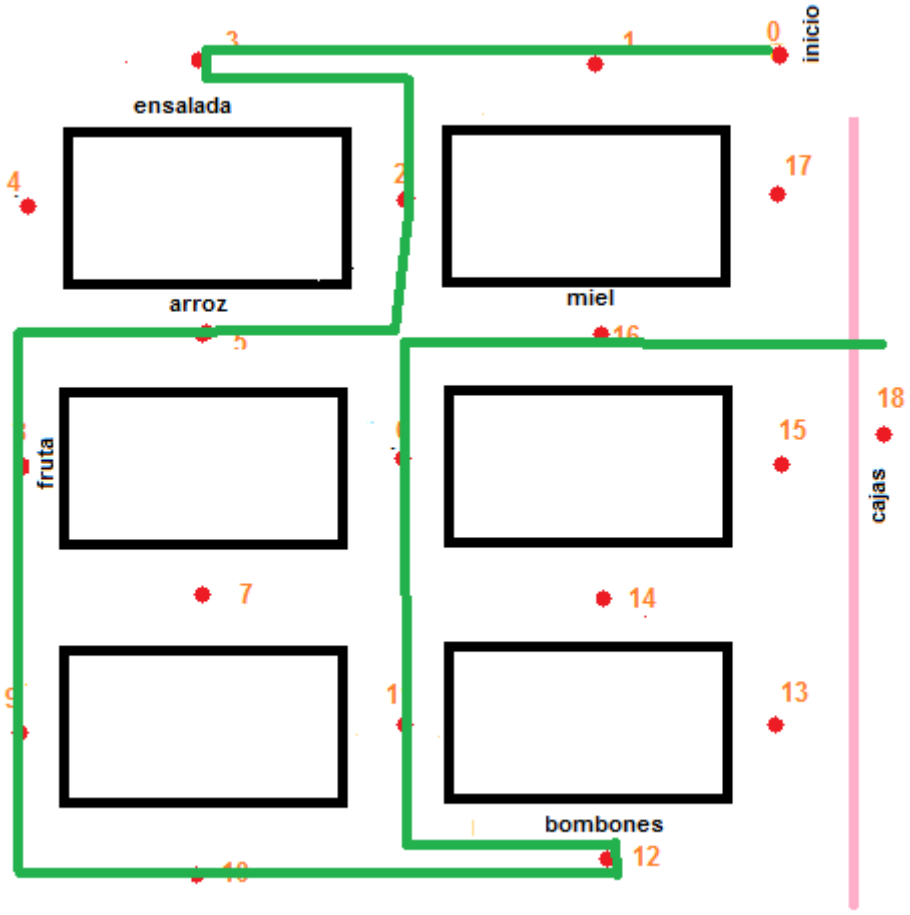

Figura 23. Ejemplo 1, algoritmo ordenado.

La distancia total recorrida con este algoritmo es de 36.

Se calcula ahora el recorrido para el algoritmo cercano y obtenemos el siguiente resultado como muestra la Figura 24:

0, 17, 16, 5, 8, 4, 3, 2, 6, 11, 12, 18

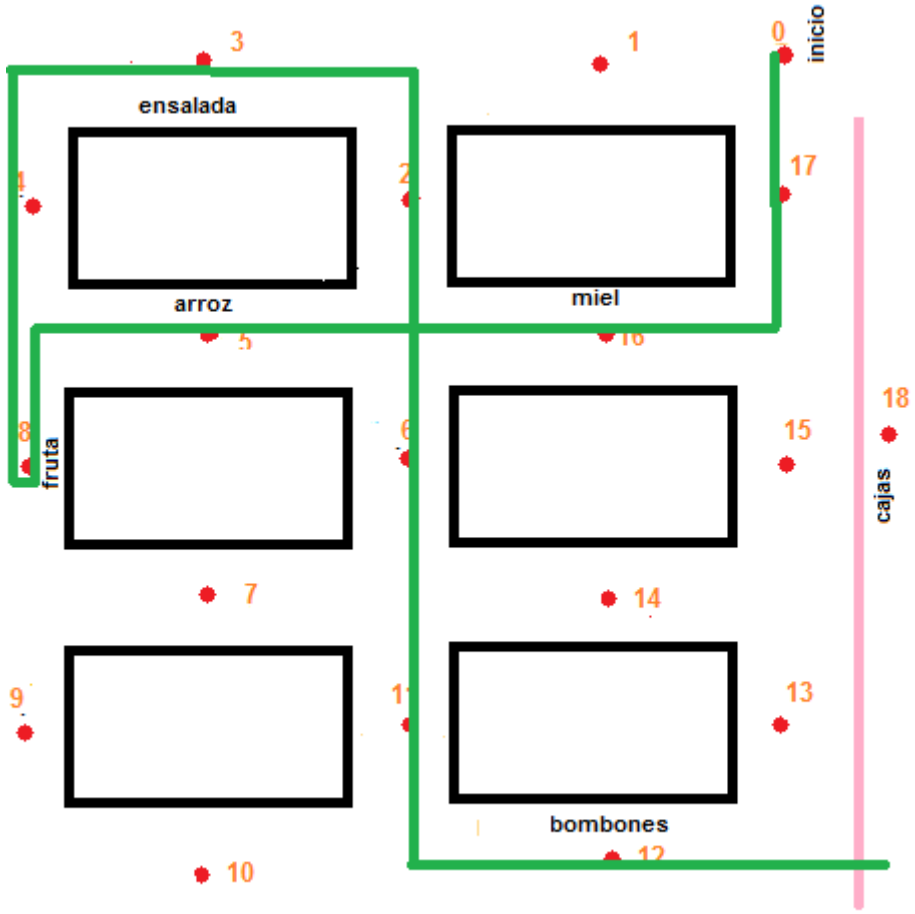

Figura 24. Ejemplo 1, algoritmo cercano.

Para el algoritmo cercano en el ejemplo 1 la distancia total recorrida es de 27,5.

Por lo tanto si se comparan ambos algoritmos aunque para muchos sea más lógico y ordenado el primer algoritmo es más eficiente el segundo algoritmo.

- Ejemplo 2.

En este ejemplo se dispone de una lista de la compra formada con los siguientes productos: cereales, legumbres y tomate.

Mediante el algoritmo ordenado se obtiene el siguiente recorrido:

0, 1, 3, 4, 5, 16, 18

Este recorrido cuenta con una distancia total de 22. En la Figura 25 se representa esta situación.

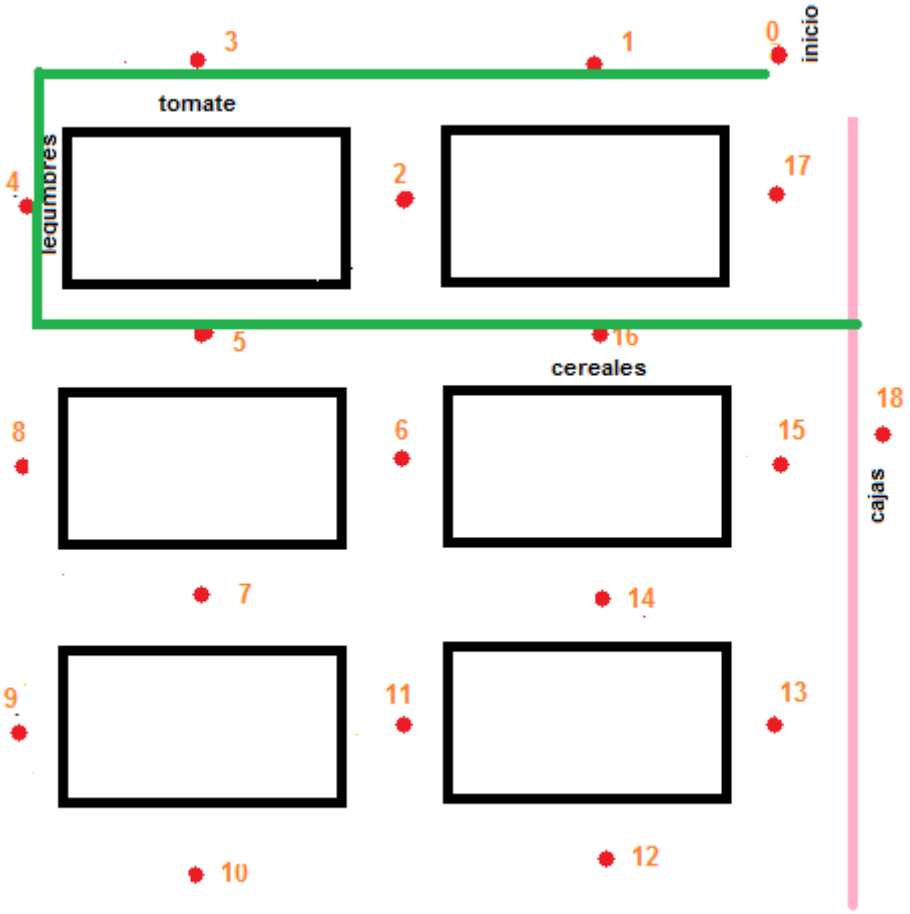

Figura 25. Ejemplo 2, algoritmo ordenado.

Por el contrario mediante el algoritmo cercano se obtiene el siguiente recorrido con una distancia de 22:

0, 17, 16, 2, 3, 4, 5, 16, 18

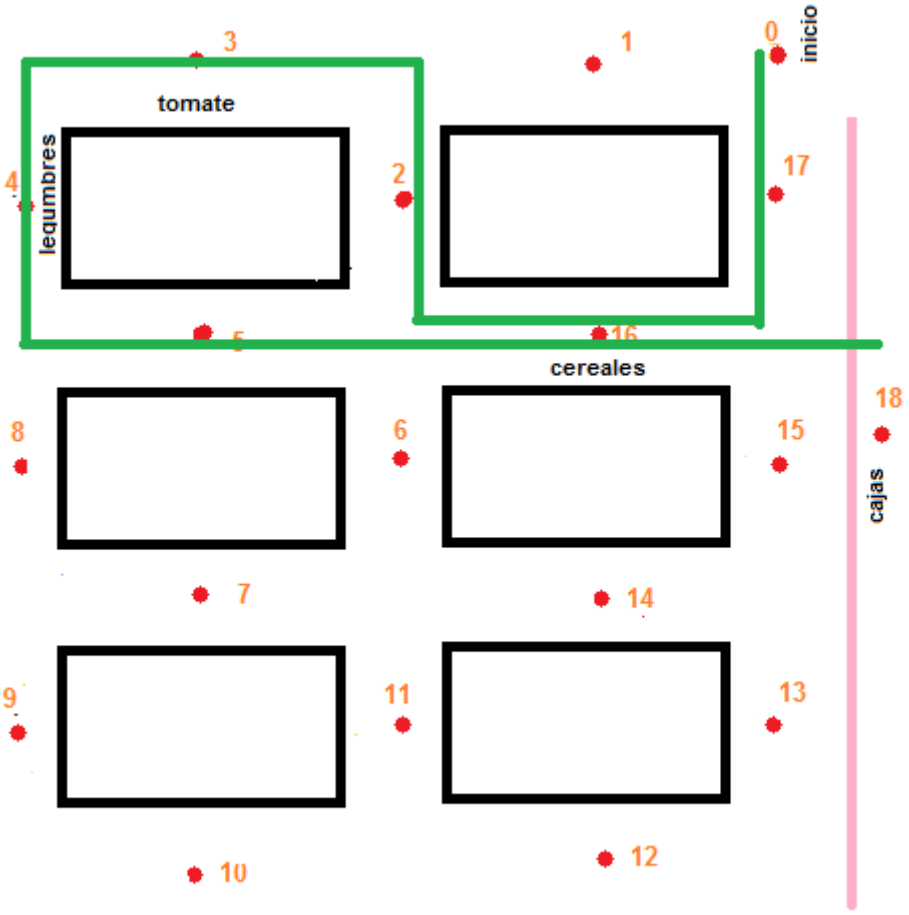

Figura 26. Ejemplo 2, algoritmo cercano.

La Figura 26 muestra el recorrido a realizar para el ejemplo 2 en el caso de utilizar el algoritmo cercano. La distancia recorrida es de 24,5 metros.

Por lo tanto si se comparan los dos resultados, en este caso con el algoritmo ordenado obtenemos una ruta de menor distancia y por lo tanto más eficiente.

Las conclusiones que podemos obtener de estos dos ejemplos es que no se puede asegurar entre los dos algoritmos cual va a ser siempre el más optimo pues dependerá de la combinación de productos elegidos de la lista de la compra. Además, a la vista de los mapas de rutas es probable, que el usuario elija uno no sólo en función de la distancia sino de sus preferencias y su percepción.

3. Implementación

En este apartado se va a explicar el contenido del proyecto Android que se ha realizado. La vista de todos los paquetes y clases es la siguiente:

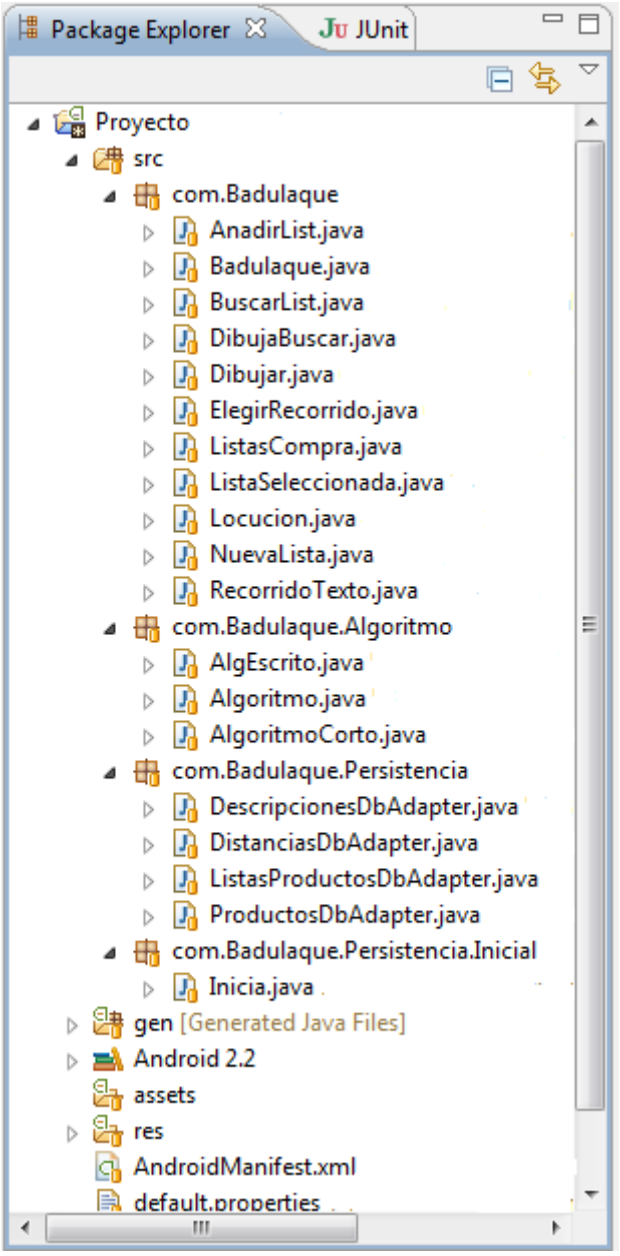

Figura 27. Vista de paquetes y clases.

3.1. Paquetes y clases

La implementación del proyecto se ha realizado en Java utilizando el entorno de desarrollo de Android para Eclipse. Se puede dividir en dos partes.

La primera parte está relacionada con el *Activity*, donde se implementa la acción de cada botón que compone cada pantalla de la que consta el programa. Una pantalla es toda la información que se puede mostrar en la pantalla del móvil de una sola vez. Cada una de estas clases de esta primera parte tiene asociado un archivo *XML* en el que se representa la configuración de la pantalla y los elementos que contiene.

La segunda parte está referida a los algoritmos y la parte relacionada con la base de datos.

Concretamente, la aplicación se compone de cuatro paquetes. Un paquete en Eclipse es una carpeta donde se encuentran las clases referidas a un mismo tema. Una clase es la estructura donde se guarda el código correspondiente a la programación de un objeto. Estos paquetes se exponen a continuación:

1) El paquete Badulaque contiene las clases que contienen las *Activities* del programa, habiendo una clase para cada pantalla que se muestra durante la ejecución de la aplicación. Dichas clases son:

   - AnadirList.java: Esta clase implementa la pantalla en la que se muestran los productos que se van a añadir o a quitar de una lista. El usuario interactúa con esta pantalla seleccionando de la lista que aparece todos los productos que desee incorporar en ella. De esta manera el usuario puede realizar su lista de la compra personalizada con el simple gesto de seleccionar los productos mediante un toque de pantalla. Esta clase puede personalizarse modificando la *base de datos* de productos que dispone el supermercado. Esta clase es un *ListActivity* que contiene un *Layout* formado a su vez por dos *Layout*. El primer *Layout* interno distribuye un botón para que ocupe toda la parte superior de la pantalla, este botón con el texto “Confirmar selección” enviará al usuario a la pantalla definida en la clase *ListaSeleccionada.java*. El segundo *Layout* contiene un *TextView* para mostrar el precio total de los productos seleccionados. Mientras que también se define un *ListView* para mostrar la lista de los productos disponibles en el supermercado.

- BuscarList.java: Esta clase se ocupa de la pantalla que da la opción de buscar un producto en el mapa. En su diseño se han utilizado tres *LinearLayout* para distribuir los dos botones y los dos *TextView* que componen el *Layout* de esta clase. Los dos botones se encuentran situados en la parte superior de la pantalla y tienen como finalidad avanzar a otra pantalla identificada con la clase *DibujaBuscar.java* o cancelar devolviendo la pantalla a la principal. Los dos *TextView* muestran un texto y permiten escribir al usuario el producto que desea buscar en el supermercado. Este *TextView* permite el autocompletado, es decir, según el usuario introduce la palabra se irán mostrando productos sugeridos que empiecen por las mismas letras que el usuario ya ha introducido.

- DibujaBusca.java: Aquí se muestra la pantalla del mapa con la ubicación de un solo elemento. Esta pantalla es mostrada después de que el usuario seleccione un producto para conocer su situación en el mapa. Esta clase es un *Activity* que contiene una View que permite dibujar en la pantalla mediante el método *OnDraw* por medio de un *canvas* proporcionado por la *View*. Un *canvas* es un objeto de la clase *Canvas* de Java que permite incluir dibujos e imágenes.

- Dibujar.java: Esta clase es la encargada de dibujar el recorrido en el mapa de una lista confeccionada por el usuario anteriormente. Esta clase es un *Activity* que contiene una *View* para poder representar el *canvas* que proporciona y así poder dibujar el recorrido sobre el plano y mostrarlo al usuario.

- ElegirRecorrido.java: Aquí se da la opción de elegir un algoritmo u otro para la representación del algoritmo. Este *Activity* contiene un *Layout* que distribuye tres botones en la pantalla. El primer botón selecciona el algoritmo denominado algoritmo ordenado, el segundo botón selecciona el algoritmo cercano y el último botón selecciona el algoritmo que mejor resultado obtenga de los dos anteriores. Cualquiera de los tres botones al seleccionarlos muestran la clase *Dibuja.javar*.

- ListasCompra.java: En la pantalla que representa esta clase se muestran las listas que ha creado un usuario con anterioridad y se encuentran grabadas en la base de datos creada para tal efecto. Esta clase es un *ListActivity* que dispone de un *Layout* para estructurar la información, en este caso el *Layout* principal engloba a otro *Layout* para alinear los dos botones de manera vertical en la parte superior de la pantalla. El primer botón con el texto "Nueva Lista" nos muestra la clase *NuevaLista.java* mientras que el segundo botón con el texto "Atrás" nos envía a la clase principal del programa. Fuera de este *Layout* interno se dispone de un *ListView* para mostrar la colección de listas guardadas y que el usuario puede elegir.

- ListaSeleccionada.java: Esta clase es la encargada de mostrar en pantalla los productos de una lista seleccionada por el usuario. Esta clase es un *ListActivity* que contiene un *Layout* formado a su vez por tres *LinearLayout*. El primer *LinearLayout* contiene tres botones con una distribución vertical en la parte superior de la pantalla. El siguiente *Layout* contiene un *TextView* para mostrar el nombre de la lista seleccionada. El último *Layout* contiene dos *TextView* para mostrar respectivamente el dinero total a pagar por los productos seleccionados en la lista y el contenido de la lista seleccionada.

- Locucion.java: En esta clase que extiende un *Activity* se implementa la locución de un recorrido, según el algoritmo seleccionado. Además posee un *Layout* estructurado en dos *LinearLayout*, el primer *LinearLayout* está compuesto por dos botones con los textos "anterior" y "siguiente" que nos permiten escuchar la anterior y siguiente locución. El segundo *LinearLayout* contiene un texto que muestra las locuciones en modo escrito.

- NuevaLista.java: En esta clase se muestra la ventana de creación de una lista nueva. Para ello se ha utilizado un *LinearLayout* que distribuye la información de esta clase a lo largo de la pantalla. Además dispone de dos *EditText* para introducir respectivamente el nombre y la descripción de la lista que se va a proceder a crear. También dispone de dos botones, el primero con el texto "Añadir productos" muestra la pantalla identificada con el nombre de la clase *AnadirList*, mientras que el segundo botón con el texto "Cancelar" muestra la pantalla de la clase con el nombre *ListaCompra*.

- RecorridoTexto.java: En esta clase que extiende un *Activity* se implementa el código que permite ver el recorrido hallado mediante una lista de acciones que indica los pasillos por donde se debe ir y que ruta se debe tomar. Para ello se toma de la clase *AlgEscrito.java* la lista de acciones que se hallan en dicha clase dependiendo del algoritmo elegido. Esta *Activity* está asociada a un *Layout* compuesto por dos LinearLayour y un *ListView*. Los *LinearLayout* en este caso sirven para enmarcar la *ListView* donde se encuentra el campo reservado para mostrar la lista de acciones.

- Badulaque.java: Esta clase contiene lo referente a la pantalla inicial de la aplicación. Cada vez que el usuario ejecute la aplicación se mostrará esta pantalla en su móvil. Consta de dos botones mediante los cuales el usuario puede avanzar a otra pantalla distinta. Si el usuario selecciona el botón primero cuyo texto es "Listas" se mostrará la pantalla correspondiente a la clase *ListasCompra.java* donde se pueden gestionar las listas de la compra. Por el contrario si el usuario pulsa el segundo botón cuyo texto es "Buscar" aparecerá la pantalla asociada a la clase *BuscarList*, en las próximas pantallas el usuario puede buscar un único producto en el supermercado.

2) En el paquete Algoritmo se encuentran las clases en las que se encuentra el código correspondiente a los algoritmos de creación del recorrido. Contiene las siguientes clases:

   - AlgEscrito.java: Esta clase es la encargada de recibir un conjunto de nodos solución que representan el recorrido a realizar y lo convierte a un conjunto de acciones representado en modo texto. Este texto hallado en esta clase se usará en las clases *RecorridoTexto.java* y *Locucion.java*.

   - Algoritmo.java: En esta clase se encuentra el código para hallar el recorrido del mapa que se ha denominado algoritmo ordenado.

   - AlgoritmoCorto.java: Aquí se implementa el algoritmo del recorrido basado en ir al siguiente punto más cercano que se ha denominado algoritmo cercano.

3) El paquete Persistencia contiene las clases relacionadas con las *bases de datos*, tanto creación como acceso a las mismas que se explican más en detalle en el siguiente apartado.

- DescripcionesDbAdapter.java: En esta clase se implementa la *base de datos* de las descripciones de cada lista.
- DistanciasDbAdapter.java: Esta clase contiene el código relacionado con la *base de datos* en la que se guardan las distancias entre los distintos puntos del recorrido.
- ListasProductosDbAdapter.java: Esta clase implementa la *base de datos* que contiene las listas con los productos que contiene cada una.
- ProductosDbAdapter.java: Aquí se implementa la *base de datos* de productos y las funciones necesarias para su manejo.

4) El paquete PersistenciaInicial contiene solamente una clase necesaria para el correcto funcionamiento de las *bases de datos*.

- Inicia.java: Esta clase se utiliza para iniciar algunas *bases de datos* al comienzo del programa. Para el correcto funcionamiento de la aplicación es necesario disponer de unos datos proporcionados por el supermercado y que se deben de incluir en las bases de datos, estos datos son las distancias entre los pasillos y los productos disponibles con su situación y precio. Para ello introducimos los datos de las conexiones y los pasillos en la *base de datos* diseñada para tal efecto *DistanciasDbAdapter* y los datos correspondientes a los productos se introducen en la *base de datos* denominada *ProductosDbAdapter*. Las otras dos bases de datos no se inicializan mediante esta clase ya que no poseen datos que procedan del exterior sino datos que son generados por el usuario mediante el transcurso de la aplicación.

3.2. Bases de datos

En las *bases de datos* se guarda toda la información necesaria para la que la aplicación pueda perdurar en el tiempo. Las *bases de datos* son creadas la primera vez que se ejecuta el programa y a partir de ese momento son accesibles cada vez que se utiliza la aplicación. Para el funcionamiento de la aplicación son necesarias cuatro *bases de datos*, que vemos a continuación:

1) Base de datos DistanciasDbAdapter: En esta *base de datos* se guarda la información relativa a las distancias de los distintos productos dentro del mapa, que se utiliza para la realización del algoritmo del recorrido. También contiene la información de puntos intermedios que serán útiles a la hora de representarlo gráficamente. Es importante resaltar que las conexiones que no estén especificadas en esta *base de datos* es debido a que no existen en la representación del mapa, es decir que si no se encuentra una pareja de nodo origen y nodo destino significa que no se pueden conectar dichos pasillos por un solo camino. Esta *base de datos* consta de los siguientes campos que la representan:

   - Origen: Este campo representa un número de nodo mediante el cual se identifica al primer elemento de la pareja que forma una conexión entre dos pasillos.
   - Destino: Identifica al segundo elemento de la pareja que forma la conexión entre dos pasillos.
   - Distancia: Distancia que hay entre el nodo origen y el nodo destino, es decir es la distancia del pasillo que comunica dichos nodos.
   - Auxx: Este campo posee un número entero que representa la coordenada horizontal para un punto auxiliar a la hora de dibujar el algoritmo en un recorrido que empiece el nodo origen y termine en el nodo destino.
   - Auxy: Coordenada vertical para el mismo punto auxiliar.

Si existe un nodo origen y un nodo destino que forma una conexión, ésta debe de ser única, ya que entre dos nodos que se comunican directamente sólo hay un posible camino entre ellos. No obstante si existe una pareja formada por un nodo origen y un nodo destino también existirá su contrario ya que las conexiones entre dos nodos son bidireccionales.

2) Base de datos ProductosDbAdapater: Esta *base de datos* contiene la información relativa a los productos que hay en el supermercado, conteniendo las coordenadas de cada producto a la hora de localizarlo en el mapa. Las coordenadas representan un punto en una imagen. Siendo la coordenada x la representación en el eje horizontal y la coordenada y la representación en el eje vertical de la imagen. Sus campos son:

- Nombre: Nombre del producto, en nuestro caso es un producto alimentario del supermercado ficticio que hemos representado.
- Coorx: Representa la coordenada horizontal de la posición del nodo donde se encuentra el producto, es utilizada para dibujar el punto de comienzo o de final de las líneas que unen los nodos que representan los pasillos.
- Coory: Representa la coordenada vertical de la posición del nodo donde se encuentra el producto, es utilizada para dibujar el punto de comienzo o de final de las líneas que unen los nodos que representan los pasillos.
- Orden: Valor que se utiliza para ver el orden del producto en la realización del algoritmo. Representa la posición que ocupa el producto en el mapa del supermercado.
- Precio: Es un número que representa el precio del producto.
- CX: Coordenada horizontal que representa el punto exacto de la estantería y el pasillo donde se encuentra el producto, se utilizada para dibujar el texto del producto en la imagen y para mostrar la situación de un producto.
- CY: Coordenada vertical que representa el punto exacto de la estantería y el pasillo donde se encuentra el producto, se utilizada para dibujar el texto del producto en la imagen y para mostrar la situación de un producto.

3) Base de datos DescripcionesDbAdapater: En esta *base de datos* es donde se guarda la descripción que el usuario realiza de cada lista. Tiene los siguientes campos:
   - Lista: Nombre de la lista.
   - Descripción: Descripción de la lista. Es un texto que se inserta para poder explicar la utilidad de dicha lista.
4) Base de datos ListasProductosDbAdapater: En esta *base de datos* se encuentran cada una de las listas creadas por el usuario y los productos que contiene cada una. Sus campos son:

- Producto: Nombre del producto.
- Lista: Lista a la que pertenece el producto.

4. Herramientas utilizadas

En este apartado se describen todas las herramientas utilizadas, tanto a nivel físico como a nivel de software. A nivel de software el programa principal con el que hemos trabajo ha sido Eclipse concretamente la versión *Helios*. Como ya explicamos anteriormente para poder trabajar en Android hemos usado el SDK de Android sobre Eclipse.

Para poder trabajar en grupo y llevar una constante actualización del proyecto hemos utilizado una herramienta que soporta un repositorio adaptado para Eclipse como es el *Subversion repository*. Un repositorio es un sitio común dónde se guardan y mantienen archivos informáticos. En nuestro caso cada miembro del grupo posee permiso para acceder y modificar el repositorio creado para nuestro fin, de tal manera que al terminar de trabajar desde un ordenador personal y actualizar el repositorio los demás componentes del grupo podían acceder sin problemas al contenido actualizado. *SVN repository* es una herramienta de *org.tigris*, una comunidad basada en crear aplicaciones de código abierto para el desarrollo de aplicaciones de forma colaborativa. Para instalarla en Eclipse simplemente en el menú *Help*, en el submenú *Install new software* se debe poner la siguiente dirección:

http://subclipse.tigris.org/update_1.6.x

con el siguiente nombre:

Subclipse1.6.xUpdateSite.

De esta manera ya podremos utilizar esta herramienta.

Después de instalar la herramienta necesitamos crearnos un repositorio, nosotros hemos elegido el repositorio que nos ofrece *Google code* totalmente gratuito. Para ello sólo hace falta registrarse y obtener una cuenta gratuita de Google. Después crearemos un repositorio mediante la opción *Project Hosting*.

Una vez que ya tenemos el repositorio y la herramienta instalada podemos trabajar en un ordenador individual y guardar el trabajo realizado en el repositorio creado. Para obtener el archivo del repositorio deberemos acceder a la perspectiva de Eclipse llamada *SVN Repository Exploring* para ello nos iremos al menú *Window* y el submenú *Open perspective*, allí elegiremos dicha perspectiva. Una vez abierta en la parte izquierda pulsaremos en el botón derecho del ratón y seleccionaremos *new*, *Reposirtoy location*, ahora nos pide la *url* donde tenemos almacenado nuestro repositorio. Esta dirección la encontramos en *Google code*, una vez que estamos dentro de nuestro proyecto accedemos al menú *Source* y la pestaña *Checkout* tal y como muestra la Figura 28 copiamos la dirección seleccionada con el circulo naranja.

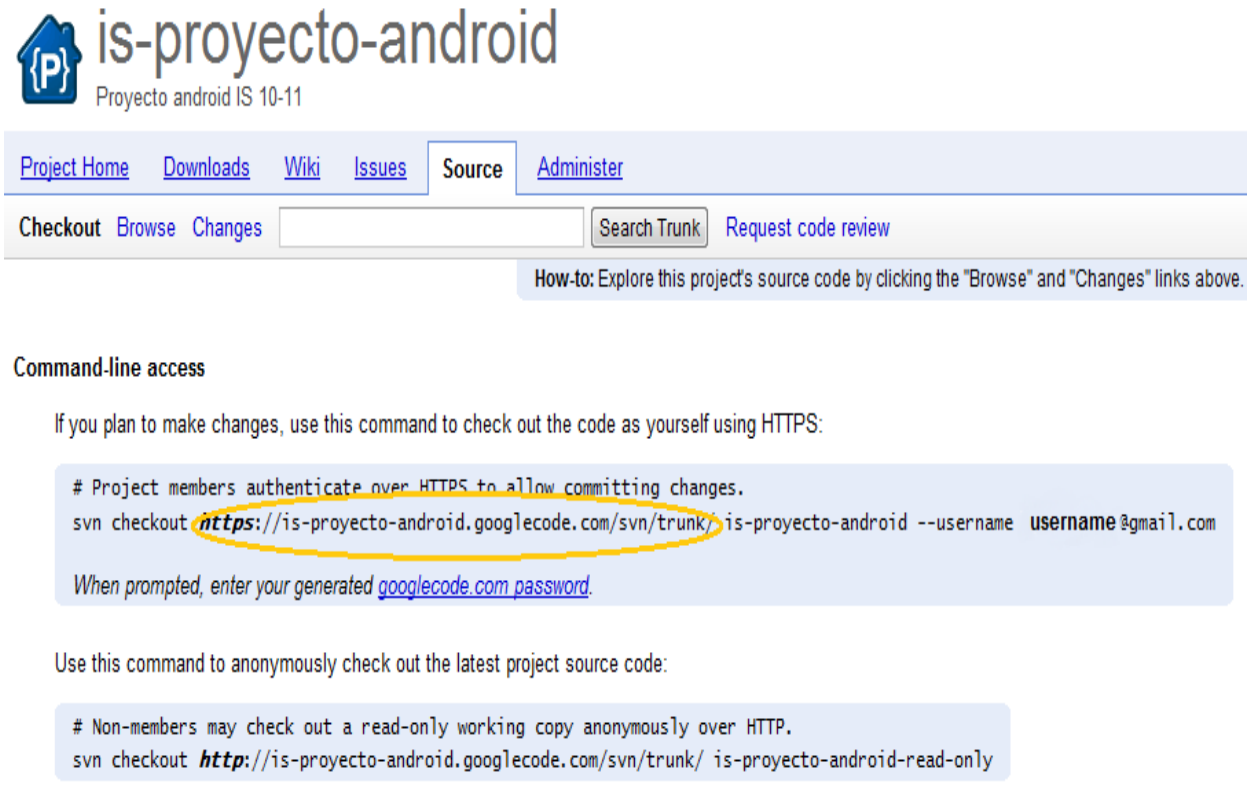

Figura 28. Google Code.

Una vez copiada la url la introducimos en la ventana de Eclipse y pulsamos *Finish*. A continuación en la parte izquierda de la pantalla nos aparecerá el nuevo repositorio creado, si pulsamos sobre él con el botón derecho del ratón y seleccionamos la opción *checkout*, se nos pedirá unos datos como el nombre del proyecto y tras aceptar obtendremos la última versión de nuestro proyecto en el área de trabajo de la *perspectiva java*. Para guardar el trabajo realizado en el repositorio en la *perspectiva java* seleccionamos el proyecto con el botón derecho del ratón y seleccionamos la opción *Team* y *Commit*.

Para poder realizar las pruebas hemos utilizado el móvil HTC Desire que cuenta con el sistema operativo Android 2.1 *Éclair*. Este dispositivo ha sido proporcionado por la Fundación General de la UCM para el desarrollo de este proyecto.

Figura 29. HTC Desire.

Con unas medidas de 119 mm de alto, 60 mm de ancho y un grosor de 11.9 mm encontramos este potente Smartphone. Cuenta con una gran pantalla de 3,7 pulgadas y en la parte inferior encontramos 3 botones principales además del cursor óptico. Estos tres botones cumplen las funciones de Home, Menú y atrás y buscar del botón doble de la derecha. Respecto al hardware cuenta con un procesador *Snapdragon a 1 GHz*, junto a *512 MB de ROM* y *576 MB de RAM*, lo que le ofrece a este dispositivo de una gran potencia para poder ejecutar varias aplicaciones a la vez y una gran rapidez para pasar de una aplicación a otra o navegar en internet. Cuenta además con una conexión *USB* para poder conectarlo al ordenador, en nuestro caso de gran utilidad para poder introducir nuestra aplicación desarrollada en este proyecto y probarla en el dispositivo.

Como características adicionales destacables hay que mencionar los siguientes sensores, el *acelerómetro*, la brújula digital, el sensor de proximidad y el sensor de luz ambiental. Para este proyecto nos ha resultado de gran utilidad el *acelerómetro*. Esta herramienta permite realizar multitud de cosas entre ellas y la que hemos utilizado para nuestro proyecto es la de girar la perspectiva de los elementos de la pantalla con tan sólo mover el dispositivo móvil en perspectiva horizontal o vertical. De esta manera podremos seleccionar la opción vertical o la apaisada en la mayoría de menús de nuestra aplicación. Otro uso de esta herramienta es en el desarrollo de los juegos permitiendo al usuario una experiencia más real, por ejemplo el juego Labyrinth en el que tienes que guiar la bola sorteando los obstáculos hasta la meta inclinando el teléfono hacia una u otra dirección. Véase Figura 30.

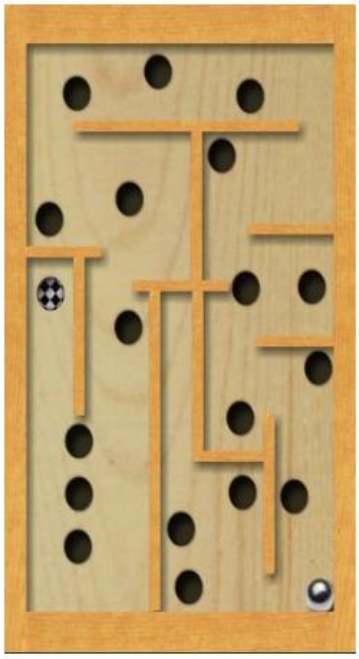

Figura 30. Labyrinth.

5. Uso de la aplicación

En este apartado se va a explicar cómo funciona la aplicación y cómo se puede navegar a través de ella.

5.1. Instalar la aplicación

Actualmente la aplicación se puede instalar desde el CD adjunto a esta memoria aunque si en un futuro la aplicación resultara comerciable se podría descargar desde una página web proporcionada por el supermercado o desde un punto Wifi en la superficie del supermercado e incluso mediante un código bidi.

En el CD se puede encontrar un archivo con la extensión .apk, este es el archivo necesario para la instalación de la aplicación en el móvil. Hay que copiar este archivo a la tarjeta de memoria del móvil o a una carpeta dentro del sistema de archivos del móvil. Una vez copiado se debe usar un gestor de archivos para poder acceder al archivo copiado y poder instalarlo como cualquier otro ejecutable en otro sistema operativo.

Una vez instalado se puede acceder a él desde la lista de programas instalados.

5.2. Ejecutar la aplicación

Para ejecutar la aplicación solamente hay que buscar el icono de la aplicación en el móvil y pulsar sobre él para que se ejecute.

5.3. Controles

Los controles que se pueden utilizar en la aplicación Shop&Go son los siguientes:

Como control principal hay que destacar la pantalla táctil, ya que a través de ella se deben seleccionar todas las opciones y menús. Además se puede usar el botón de menú del móvil para hacer aparecer un menú emergente en la pantalla donde se muestran opciones alternativas como salir a la pantalla inicial. Otro botón que se puede utilizar es el botón de volver que dispone el móvil, este botón vuelve a la pantalla anterior salvo en las pantallas en las que es necesario guardar definitivamente un dato en las que esta opción no está disponible.

5.4. Manual de usuario

En este apartado se explican todos los pasos para poder navegar a través de la aplicación y entender todas sus funciones.

Al iniciar la aplicación Shop&Go en el móvil se puede ver la siguiente pantalla que aparece en la Figura 31:

Figura 31. Pantalla inicial.

Las dos funciones principales de la aplicación Shop&Go son mostrar el recorrido de una lista determinada y buscar un único producto en el mapa. Aunque esas son las dos funciones principales Shop&Go permite hacer otras muchas funciones que se explican a continuación.

- Creación de una nueva lista.

Para crear una lista de la compra, desde la pantalla inicial hay que pulsar el primer botón *Listas*. Al pulsar dicho botón aparece la siguiente pantalla que se muestra en la Figura 32.

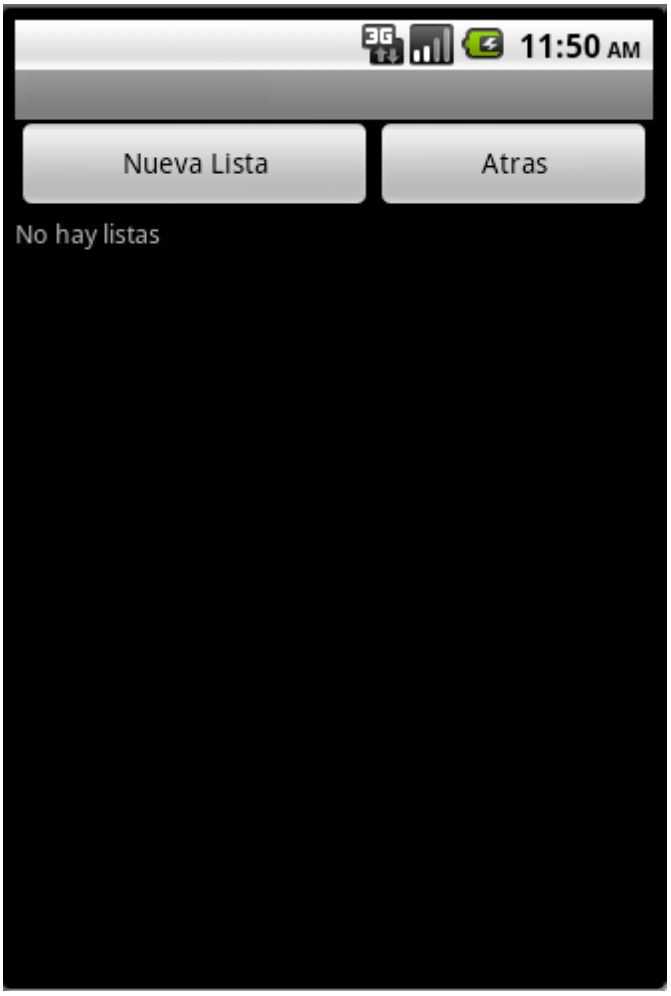

Figura 32. Pantalla que muestra las listas.

En esta pantalla se muestran las listas que hay actualmente guardadas, en la Figura 32 no existe ninguna lista creada. Para crear una lista nueva hay que pulsar el botón *Nueva Lista* y aparecerá la interfaz que se muestra en la Figura 33.

Figura 33. Creación de una lista.

En esta pantalla se dispone de varios campos a rellenar. Es obligatorio introducir un nombre para la nueva lista que se va a crear, en cambio no es obligatorio introducir una descripción que explique el uso de la misma. Una vez introducidos los campos se pulsa el botón *Añadir productos*, que mostrará la interfaz de la Figura 34.

Figura 34. Elección de los productos.

En esta pantalla se muestran todos los posibles productos que se encuentran en el supermercado. Hay que pulsar sobre el producto para seleccionarlo, en el momento en el que este seleccionado aparecerá en la parte derecha de la fila del producto un símbolo de un tick en verde tal y como muestra la Figura 35.

Figura 35. Selección de un producto.

Se puede desplazar con el gesto de *drag and drop* la lista hacia abajo y hacia arriba para buscar los productos deseados. De esta manera se confecciona la lista que se está creando, una vez seleccionados todos los productos hay que pulsar el botón de la parte superior de la pantalla *Confirmar selección*. Una vez pulsado dicho botón se mostrará la siguiente pantalla en el móvil que muestra la Figura 36.

Figura 36. Contenido de la lista en creación.

En la pantalla de la Figura 36 se encuentran en la parte central los productos que forman parte de la lista que se acaba de crear con el título que aparece con un letrero en verde. Si la lista es muy grande se puede desplazar en ella con el gesto del *drag and drop*. Además muestra el precio de cada producto y el precio en total de la lista.

- Visualización de todas las listas disponibles.

Shop&Go permite almacenar una gran cantidad de listas de la compra para su futuro uso. Es normal realizar una misma compra o similar a lo largo del tiempo, por ejemplo la lista de todos los lunes, de cada quince días o de cada mes. Para ver las listas disponibles desde la pantalla inicial que muestra la Figura 31 se debe pulsar el primer botón *Listas*, entonces aparecerán todas las listas que anteriormente ha creado el usuario de la aplicación.

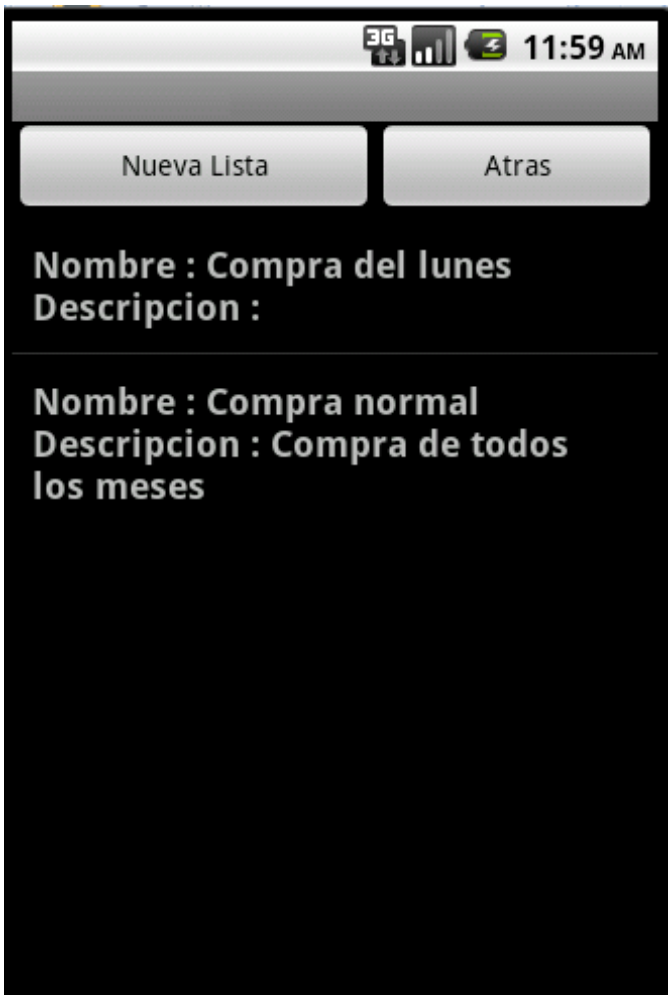

Figura 37. Visualización de las listas disponibles.

- Borrar una o todas las listas.

Para eliminar de nuestra aplicación las listas actuales desde la pantalla inicial pulsamos el primer botón que hace referencia a todos los datos de las listas. A continuación podemos ver las listas disponibles, para borrar una única lista hay que situarse sobre ella y mantener el pulsado durante unos segundos. En cambio para borrar todas las listas se debe pulsar en el botón *menú* del móvil y a continuación seleccionar el botón de *Borrar listas* como muestra la Figura 38.

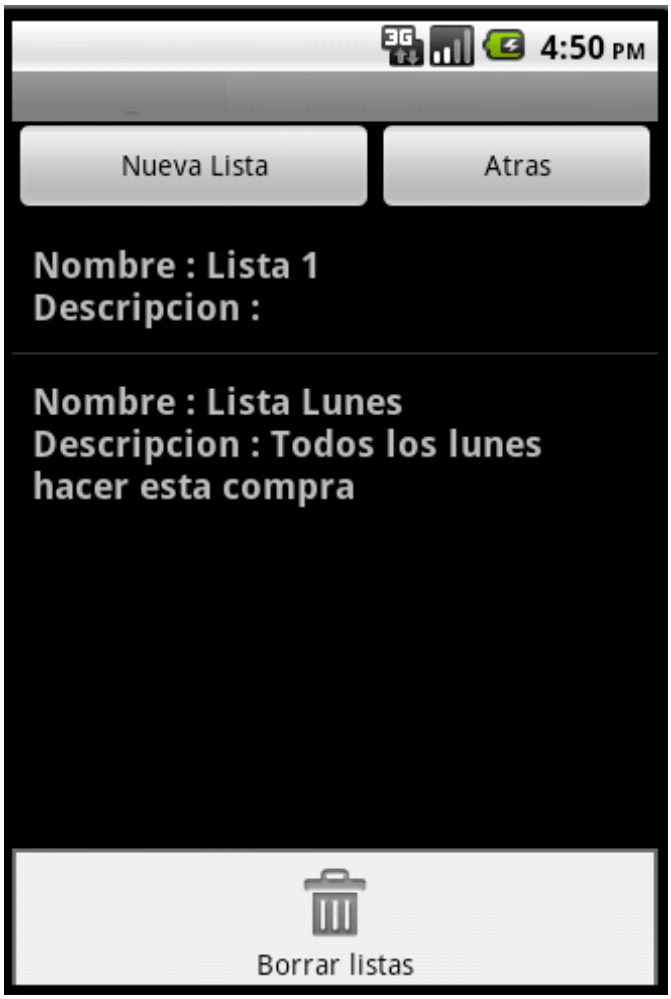

Figura 38. Borrado de todas las listas disponibles.

- Edición de una lista.

Shop&Go permite editar fácilmente cualquier lista que se encuentre creada. Para ello desde la pantalla inicial que muestra la Figura 31 se debe seleccionar el primer botón *Listas*, a continuación aparecerán todas las listas disponibles y el usuario tanto para ver su contenido como para editar una de ellas deberá seleccionarla. Aparecerá algo similar a la Figura 37, donde existen dos listas creadas.

Al seleccionar una lista aparecerá una pantalla con todo el contenido de la lista similar a la pantalla que muestra la Figura 36. Para editarla hay que pulsar el botón superior *Modificar* y se mostrará una pantalla como la Figura 35 donde se muestran los productos incluidos en la lista mediante un símbolo de un tick verde. Para borrar un producto de la lista simplemente hay que pulsar sobre el y la marca desaparecerá. Por el contario para incluir más productos hay que pulsar sobre ellos y la marca aparecerá. Una vez realizado los cambios pulsar el botón *Confirmar selección*.

- Mostrar el recorrido de una lista mediante una imagen.

Si se parte de la pantalla inicial se selecciona el primer botón *Listas*, a continuación se selecciona la lista de la cual se desea ver el recorrido o se crea una nueva. Una vez en la pantalla que muestra el contenido de la lista que muestra la Figura 36 hay que pulsar el botón *Ver recorrido* y entonces aparecerá la pantalla que muestra la Figura 39.

Figura 39. Elección del algoritmo.

A continuación se muestra una pantalla en la que se pueden elegir tres opciones distintas. Hay que elegir el algoritmo mediante el cual se quiere que se calcule el recorrido que se va a realizar en el supermercado. Una vez pulsado uno de los tres botones nos mostrará una imagen como la de la Figura 40.

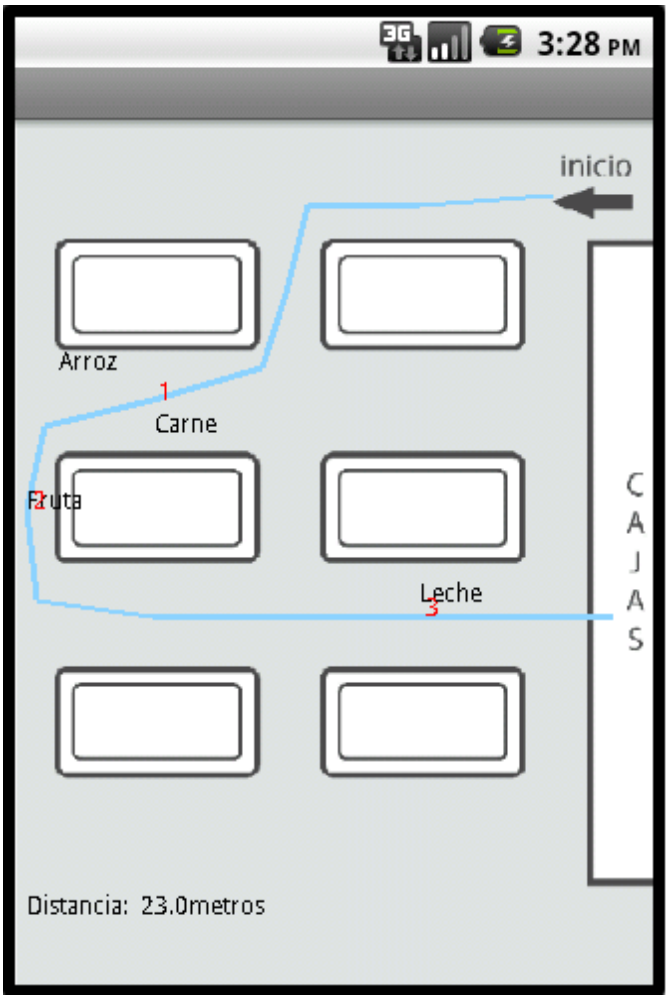

Figura 40. Recorrido de la lista creada.

En la pantalla que se muestra en la Figura 40 se representa mediante una imagen el recorrido que hay que realizar en el supermercado para comprar la lista de la compra que se ha seleccionado. En la imagen se representa el camino a recorrer por medio de una línea azul y los elementos de la lista se encuentran situados en el mapa, además se encuentran numerados los productos en el orden que se deben de coger. Además se muestra en la parte inferior de la pantalla la distancia a recorrer para realizar el recorrido propuesto. Para salir de esta pantalla hay que pulsar el botón *menú* del móvil y a continuación el botón *salir* en el menú emergente que muestra la Figura 41.

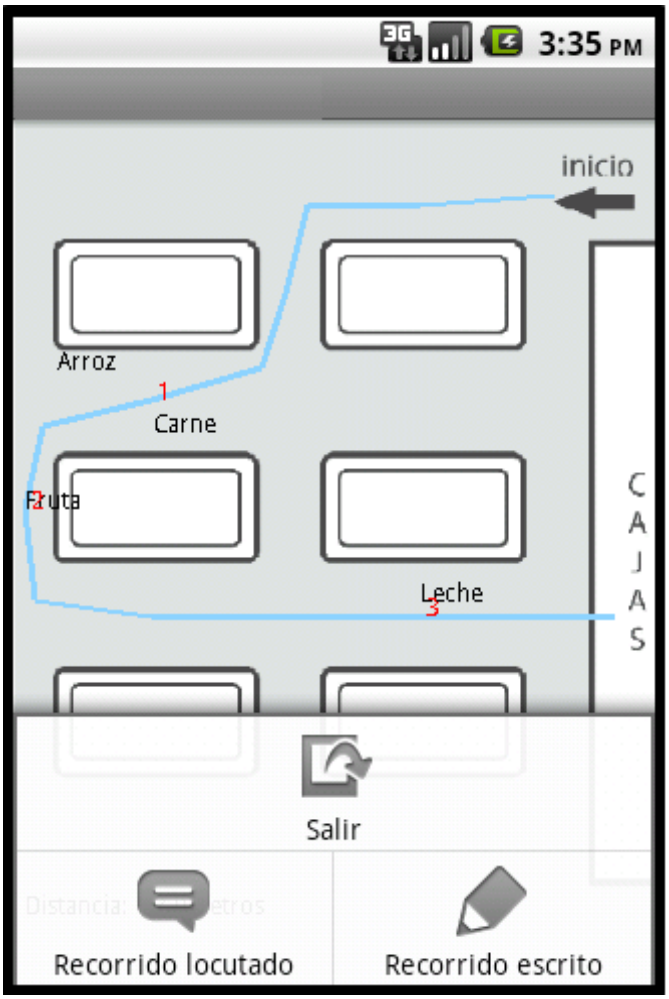

Figura 41. Menú de opciones.

- Mostrar el recorrido de una lista mediante un texto o una locución.

Para poder ver en la pantalla del móvil la serie de instrucciones que hay que realizar para coger los productos de una determinada lista hay que seguir los mismos pasos que para mostrar el recorrido en imagen y una vez que se llegue a la pantalla que representa la Figura 40 donde podemos ver el mapa con una imagen hay que pulsar el botón de *menú* del móvil para que aparezcan una serie de opciones como muestra la Figura 41, a continuación seleccionar bien el botón *Recorrido locutado* para poder escuchar las instrucciones a seguir o el botón *Recorrido escrito* para poder leer en la pantalla las instrucciones.

En la Figura 42 se puede ver la pantalla en el caso de haber seleccionado la opción de recorrido escrito.

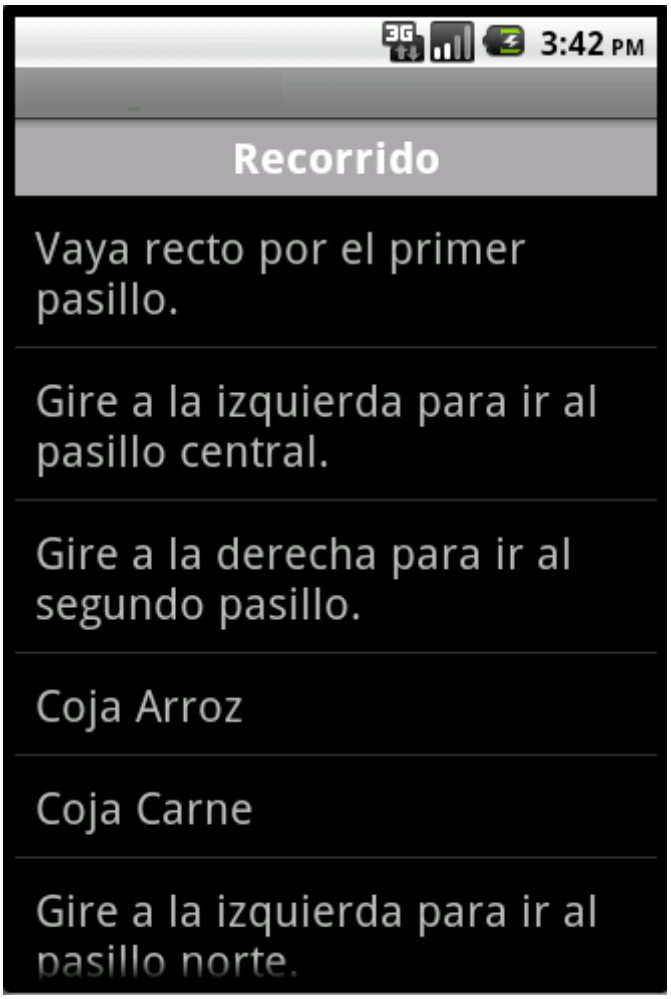

Figura 42. Recorrido escrito.

En cambio la Figura 43 muestra la pantalla de haber seleccionado el botón *Recorrido locutado*.

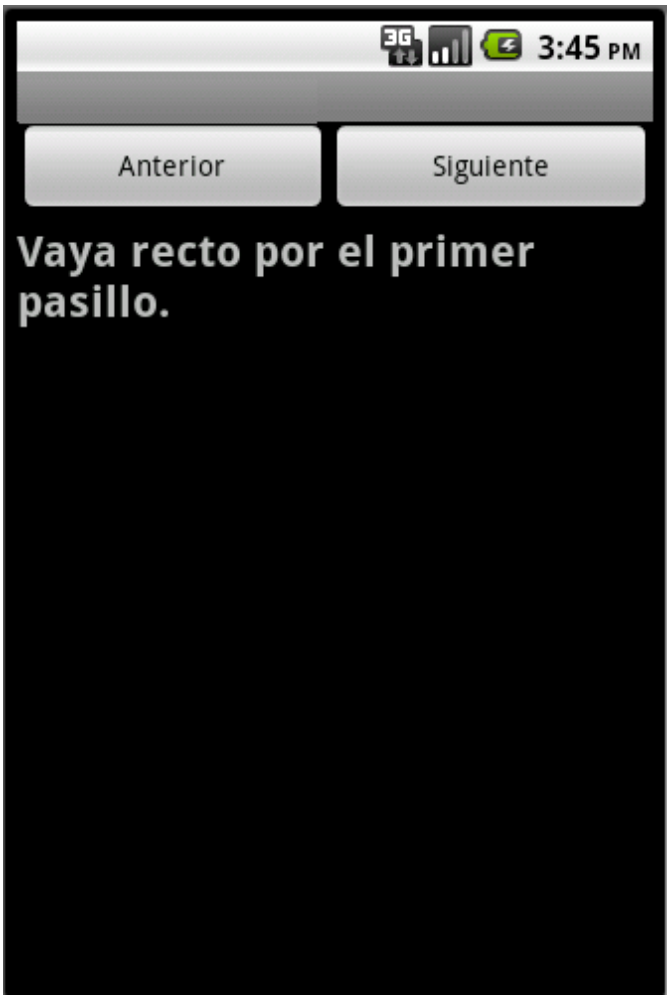

Figura 43. Recorrido locutado.

En la pantalla del recorrido locutado las instrucciones aparecerán de una en una mientras se escucha el texto con una voz por el altavoz que dispone el teléfono móvil. Para ver y escuchar la siguiente o la anterior instrucción hay que pulsar los botones *Anterior* o *Siguiente*. En ambos recorridos cuando se quiera salir de la pantalla hay que pulsar el botón *menú* del teléfono móvil y aparecerá un menú en la pantalla con la opción de salir tal y como muestra la Figura 44.

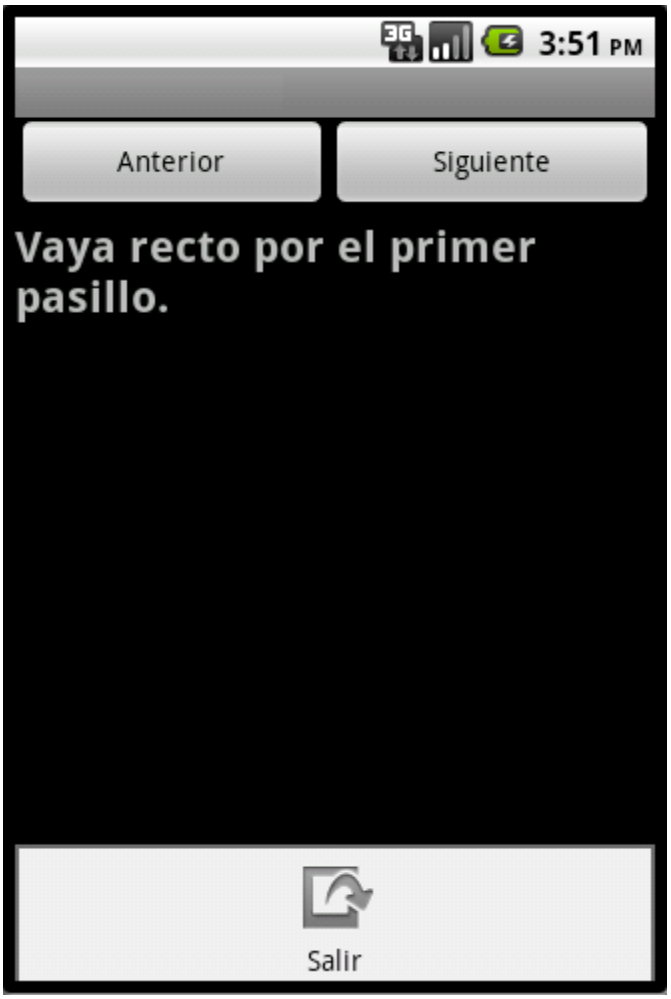

Figura 44. Menú salir.

- Seleccionar un producto en la imagen del recorrido

Si desea ir seleccionando los productos que va cogiendo de las estanterías con solo pulsar en la pantalla encima del texto que especifica el producto se pintará un círculo a su alrededor.

- Mostrar la situación de un determinado producto.

A continuación se muestra cómo navegar en la aplicación Shop&Go para mostrar la situación de un producto en el supermercado.

Desde la pantalla inicial hay que pulsar el segundo botón *Buscar* un producto como muestra la Figura 31. Al pulsar dicho botón aparecerá la interfaz de la Figura 45 donde hay que introducir el nombre del producto a buscar. El campo diseñado para tal efecto mostrará según se escribe palabras sugeridas que empiecen por las letras que se introducen aportando al usuario una mayor rapidez.

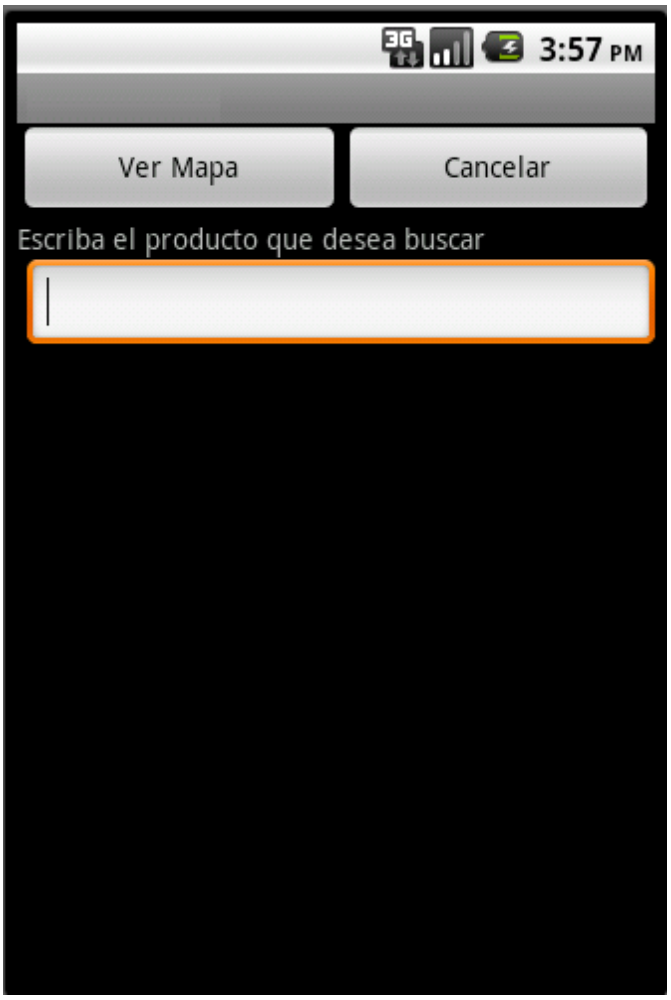

Figura 45. Búsqueda de un producto.

Para ver la situación del producto introducido pulsar el botón *Ver Mapa* y se mostrará una pantalla similar a la de la Figura 45, dónde por medio de una imagen se representa la localización del producto. El punto exacto se encuentra marcado con un punto verde. Para ir a la pantalla principal pulsar el botón *menú* del móvil y seleccionar el botón *Salir*. La Figura 46 muestra la localización de un determinado producto.

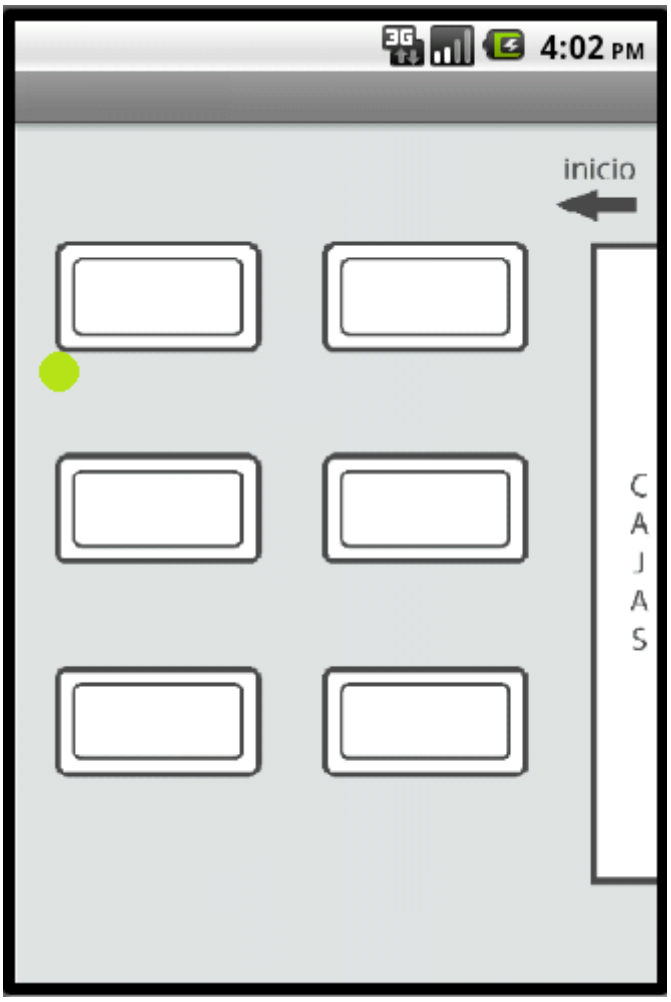

Figura 46. Localización del producto.

- Enviar por correo electrónico una lista de la compra.

Para poder realizar está acción desde la pantalla inicial del móvil hay que pulsar el primer botón *Listas* y aparecerán todas las listas disponibles. Se puede seleccionar una de ellas o crear una nueva para enviarla por correo, de todas maneras se selecciona la lista deseada y aparecerá una pantalla similar a la Figura 36. Hay que pulsar el botón *menú* del móvil y aparecerá un menú emergente como muestra la Figura 47.

Figura 47. Opción enviar por e-mail.

Al pulsar sobre el botón *Enviar e-mail* se abre la aplicación configurada en el móvil como servicio de correo electrónico con un nuevo correo redactado. En este correo aparece como asunto del e-mail “Lista de la compra” y en el campo de texto aparece la lista de la compra con el total a pagar. Por último aparece como firma del correo electrónico “Lista hecha por la aplicación Shop&Go”. Solamente faltaría introducir el destinatario del correo y pulsar el botón *Enviar* de dicha aplicación.

5.5. Diagramas

Para entender correctamente la división del proyecto en distintas clases de representan a continuación diversos diagramas.

- Diagrama de conexiones entre clases

Este diagrama representa mediante líneas la conexión entre dos clases distintas, esto significa que se puede navegar mediante un botón de la pantalla de una a otra. Este diagrama se muestra en la Figura 48, solo muestra las clases del paquete Shop&Go ya que es el que tiene las clases que representan Activity.

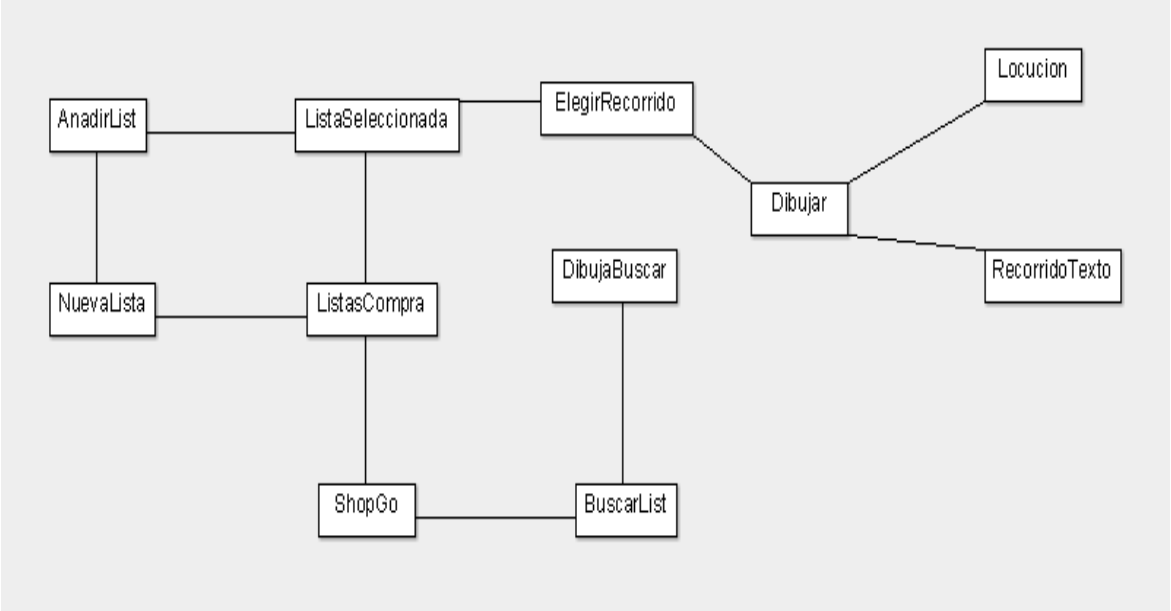

Figura 48. Diagrama de conexiones.

- Diagrama de clases

Este diagrama representa la estructura de la aplicación mostrando sus clases, atributos y las relaciones entre ellos.

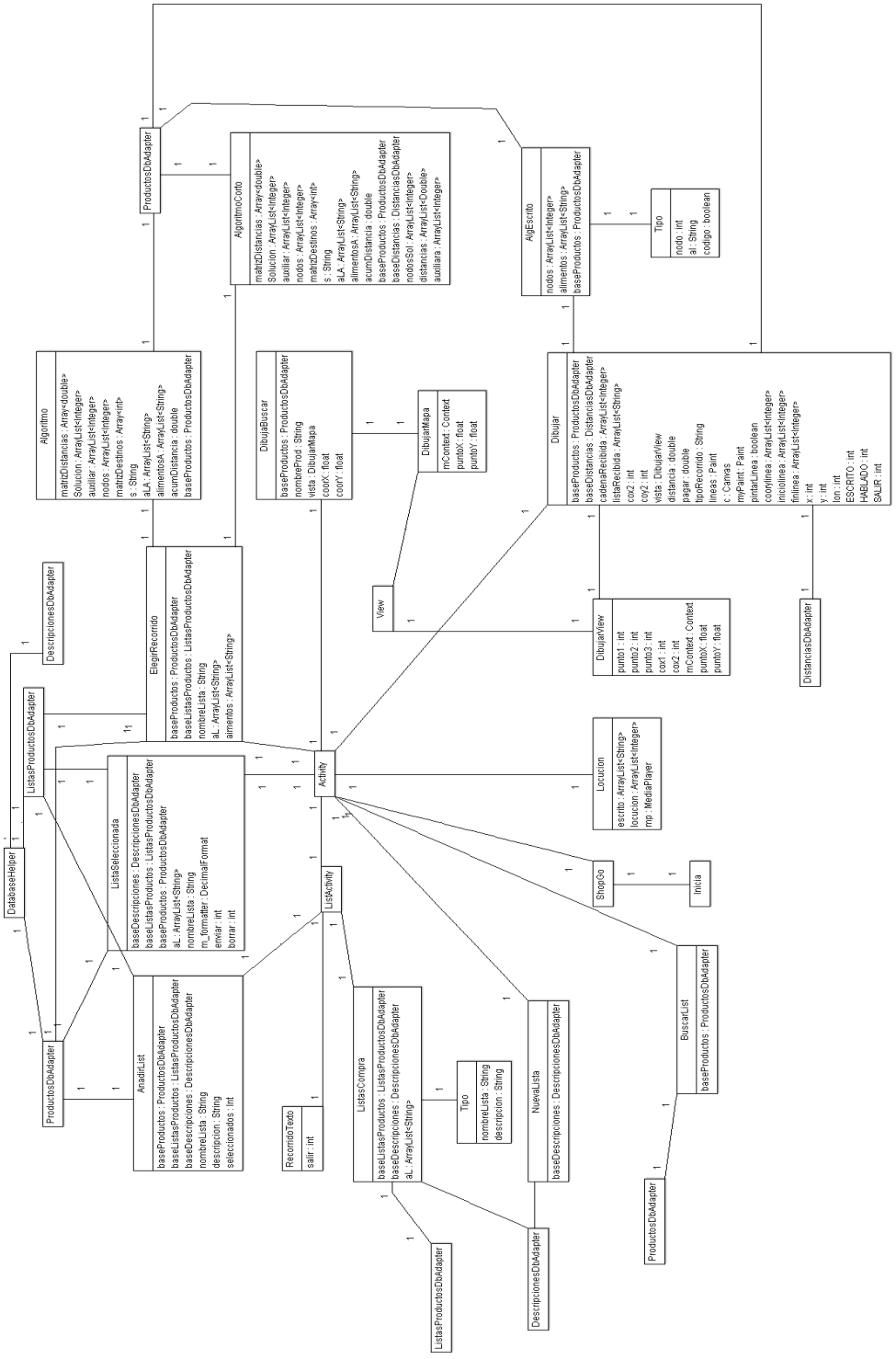

Figura 49. Diagrama de clases

- Diagrama de casos de uso

Este diagrama muestra los distintos casos de usos para el único actor en esta caso el usuario de la aplicación Shop&Go.

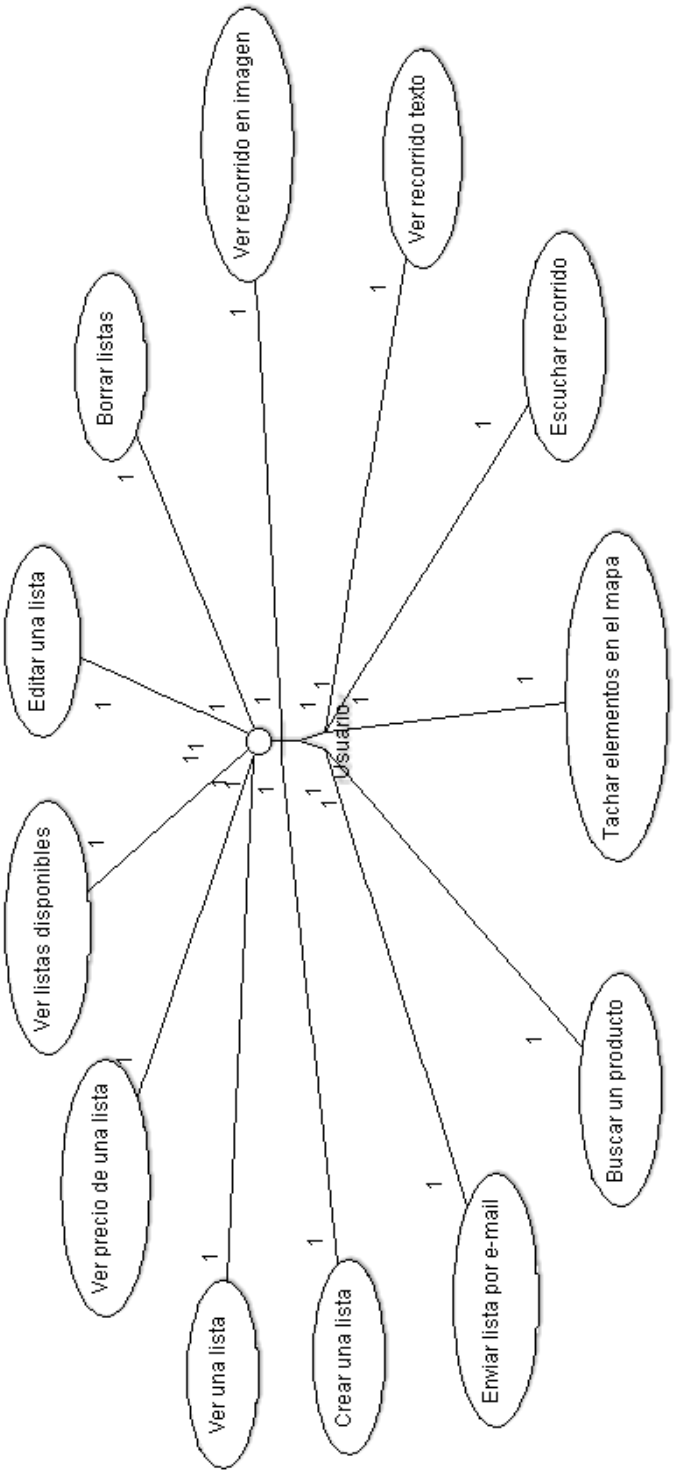

Figura 50. Diagrama de casos de uso.

Capítulo 5

Opiniones de los compradores de supermercados

1. Opiniones

Con el fin de justificar el uso y comercialización futura de la herramienta Shop&Go, hemos decidido realizar un estudio de la opinión de los clientes potenciales de dicho producto. Para ello se han seguido los pasos básicos de un pequeño estudio de mercado: 1. Realización de encuestas; 2. Almacenamiento y depuración de los datos y 3. Análisis de los datos.

En la fase de realización de las encuestas y en primer lugar, hemos diseñado una encuesta sencilla con el fin de obtener respuestas rápidas y veraces sin molestar a los encuestados demasiado tiempo para no deteriorar la calidad de las respuestas. El sector poblacional al que se ha aplicado la encuesta ha sido compradores de una gran superficie dentro de un centro comercial. La encuesta se ha realizado a las personas que se dirigían a hacer la compra (entrada al supermercado). Los detalles pueden consultarse en el Apéndice número 5.

Se han realizado 40 encuestas, lo que supone un tamaño muestral no muy grande pero descriptivo y suficientemente homogéneo para el problema que queremos analizar. A partir del análisis de estos datos se han podido obtener los siguientes resultados y conclusiones.

Pregunta 1: Realización de la lista de la compra. Las personas han contestado la frecuencia con la que llevan una lista de la compra, evaluando los datos hemos obtenido el gráfico de la Figura 51.

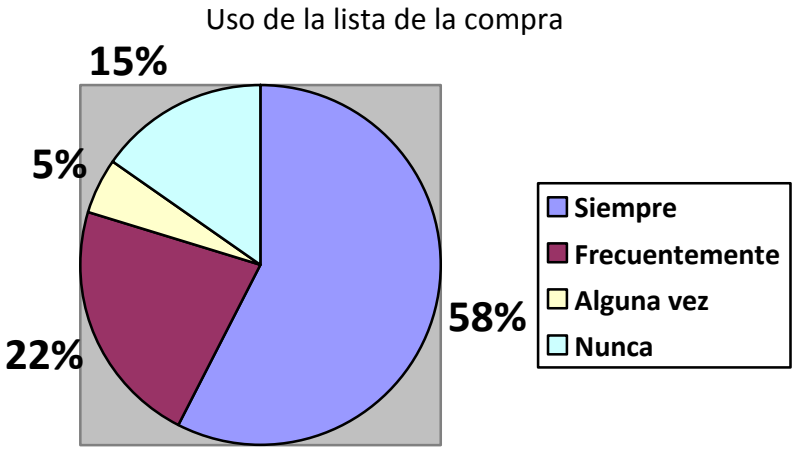

Figura 51.Gráfica de resultados.

Pregunta 2: Utilidad del teléfono móvil para realizar la lista de la compra. Otra pregunta realizada ha sido si utilizarían el móvil para llevar apuntada la lista de la compra. El resultado obtenido fue un 20% que dijeron que preferían llevarla escrita en papel y un 80% que opinó que la llevarían en el móvil. La Figura 52 muestra un gráfico de esta situación.

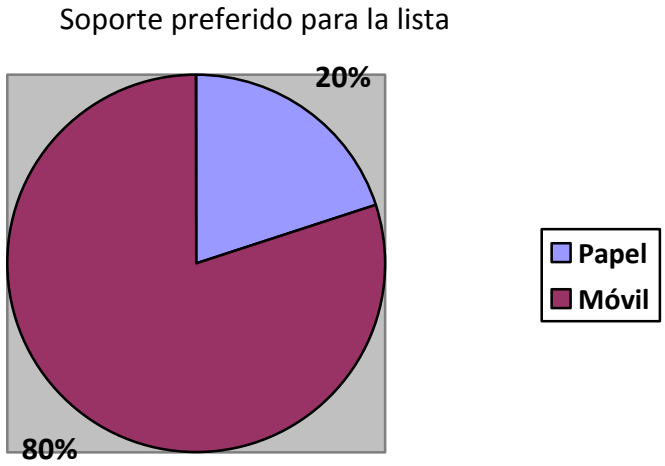

Figura 52. Gráfica de resultados.

Pregunta 3: Deseo de conocer con antelación el coste de la lista de la compra. En cuanto a la pregunta de si les gustaría conocer con anterioridad el coste de su lista de la compra han respondido lo representado en la Figura 53.

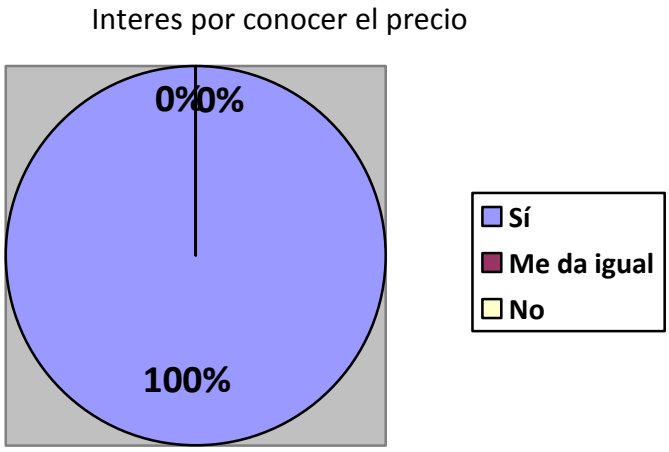

Figura 53. Gráfica de resultados.

Pregunta 4: Sistema Operativo utilizado. El sistema operativo más usado ha sido Android con un 34%. En la Figura 54 podemos ver la distribución de los sistemas operativos usados entre nuestros encuestados.

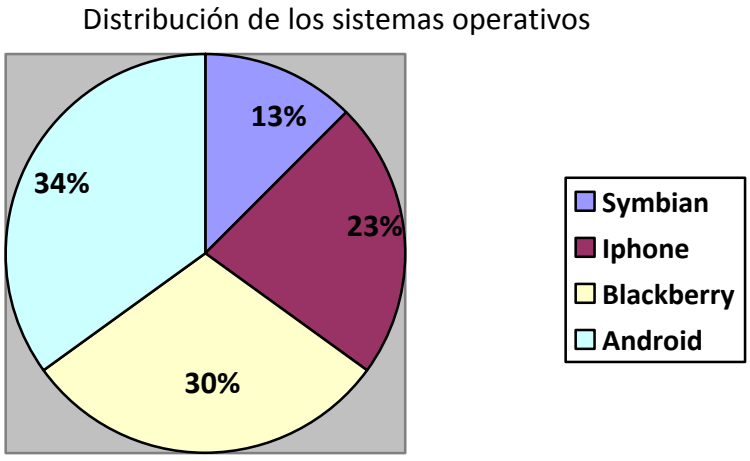

Figura 54. Gráfica de resultados.

Pregunta 5: Interés sobre herramientas de optimización de rutas en supermercados. Respecto a la pregunta de si le resulta interesante usar una aplicación móvil para facilitar la compra en un supermercado hemos obtenidos los resultados de la Figura 55.

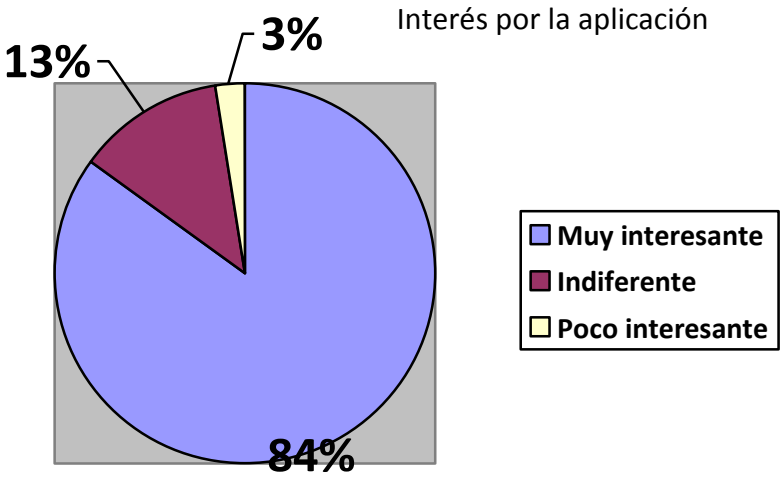

Figura 55. Gráfica de resultados.

Pregunta 6: Valoración de Shop&Go. Por último los encuestados nos respondieron a su valoración de nuestra aplicación después de explicarles y mostrarles nuestro trabajo en el móvil de pruebas. Los resultados han sido bastantes favorables ya que la nota media de nuestro proyecto ha sido un 4,35 sobre 5.

Gracias a esta encuesta hemos podido conocer las opiniones de los compradores de supermercados y afirmar con seguridad que nuestra aplicación sería una segura inversión para cualquier supermercado que estime a sus clientes.

Capítulo 6
Conclusiones y trabajos futuros

1. Conclusiones

Se ha desarrollado una aplicación para dispositivos móviles que consigue facilitar el proceso de compra en un supermercado por parte de un usuario. Utilizando la aplicación ahorrará tiempo a la hora de realizar sus compras y evitará deambular por la superficie en busca de uno o varios productos.

La aplicación ha sido desarrollada para dispositivos táctiles de telefonía móvil, un sector de la tecnología cada vez más al alcance de los usuarios y cuyo uso está creciendo en gran medida los últimos años, lo que facilitará que el programa sea accesible para un gran numero de usuarios. Esto se ve reforzado por la posibilidad de poder obtener el recorrido a realizar de manera hablada, lo que supone una gran aportación para los usuarios invidentes.

Ya existen en el mercado aplicaciones para crear listas, pero ninguna muestra la ruta a realizar por el supermercado, así que se ha conseguido llegar un paso más allá en este tipo de aplicaciones, añadiendo una funcionalidad que los desarrolladores consideran muy útil gracias al ahorro de tiempo que supone su utilización.

2. Trabajos futuros

La aplicación desarrollada puede ser ampliada en varios aspectos. A continuación aparecen posibles mejoras que se podrían introducir en el programa:

- Una de las principales funcionalidades a añadir debería ser la posibilidad de tener almacenados en el dispositivo móvil los mapas de diferentes supermercados y las ubicaciones y precios de sus productos.
- El apartado anterior sería complementado con la posibilidad de descargar nuevos mapas o actualizar los disponibles y su información, bien a través de internet o bien a través de la conexión Bluetooth de los teléfonos móviles al llegar al centro comercial.

- Dada la diversidad de sistemas operativos que existen para dispositivos móviles debería implementarse la aplicación para otros sistemas además de Android, como el iOS de Iphone o Symbian de Nokia, para que sea accesible al mayor número de usuarios posible.
- Otro aspecto que podría mejorarse son los algoritmos de creación de la ruta a seguir por parte del usuario, para que fueran más eficientes y/o precisos a la hora de encontrar la ruta óptima. También se podría hacer que la ruta fuera a los productos directamente en vez del pasillo donde se encuentran, opción desechada por temas de rendimiento, pero que tal vez podría realizarse al mejorar los algoritmos.
- Una opción muy interesante sería implantar un GPS interno dentro del supermercado, de tal manera que un móvil con conexión GPS se pudiera guiar a través de los pasillos de manera interactiva.
- Además podría implementarse la posibilidad de importar al programa una lista creada en la aplicación por otro usuario y que hubiera sido enviada vía e-mail, y a partir de esa lista crear la ruta.
- Para hacer la aplicación más interesante para las empresas y que tuvieran interés en implantarlas en sus superficies, podría añadirse la opción de mostrar al usuario ofertas y promociones que haya en cada momento en el supermercado.
- Por último se podría añadir alguna mejora a la interfaz en forma de opciones de personalización, como poder cambiar la imagen inicial o los colores de la aplicación, aunque esto es menos relevante.

Capítulo 7
Apéndices

Apéndice 1. Instalación Eclipse

Para instalar Eclipse en su equipo debe descargar el entorno de desarrollo, para ello hay que acceder a la siguiente página web:

http://www.eclipse.org/downloads.

En esta página web aparece una pantalla con todas las versiones de desarrollo Eclipse disponibles. En nuestro caso utilizamos Eclipse *Classic*.

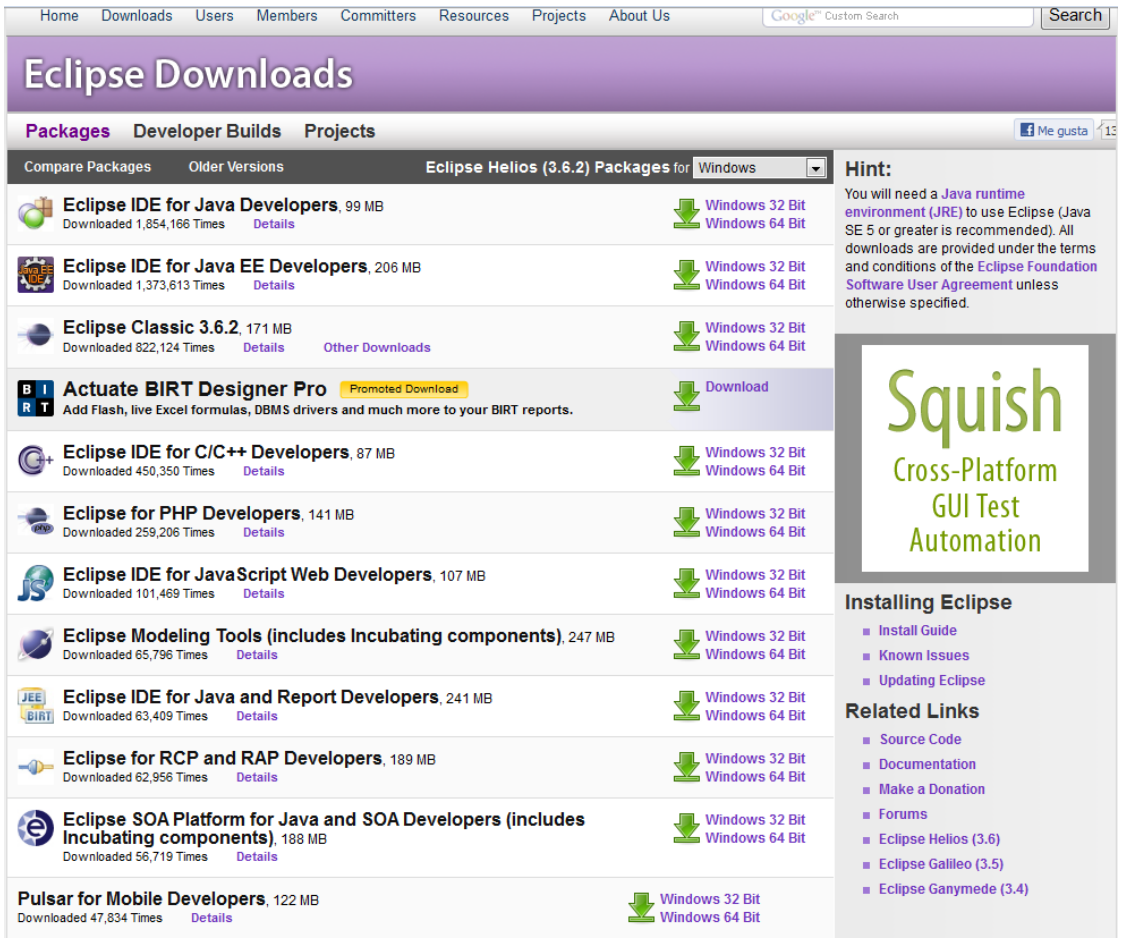

Figura 56. Versiones de Eclipse.

El archivo descargado es un archivo comprimido. El siguiente paso es acceder al archivo y descomprimir todos sus datos en una carpeta con el mismo nombre. Una vez descomprimido se obtiene el entorno de desarrollo, no es necesario ninguna instalación. Para ejecutar el entorno Eclipse simplemente se accede a través del archivo Eclipse.exe dentro de la carpeta que hemos creado con los datos descomprimidos. Como consejo crear un acceso directo a este archivo por ejemplo en el escritorio para un rápido acceso al programa.

Para ejecutar el entorno Eclipse como hemos dicho antes se pulsa en el archivo ejecutable que se encuentra en la carpeta extraída llamado eclipse.exe, que abrirá el entorno de desarrollo, y solicitara una ruta por defecto donde se guardarán los nuevos proyectos que se vayan creando. Se podrá marcar la opción para utilizar siempre la ruta introducida y que no se solicite cada vez que se abre Eclipse:

Una vez iniciado Eclipse, le solicitará un directorio de trabajo en el que guardar todos los proyectos que realice. Puede cambiar el valor por defecto a cualquier carpeta de su sistema, una vez configurado el espacio de trabajo le dará la opción de usar siempre el mismo y no preguntar más o en cambio cada vez que inicie Eclipse poder configurar esta opción.

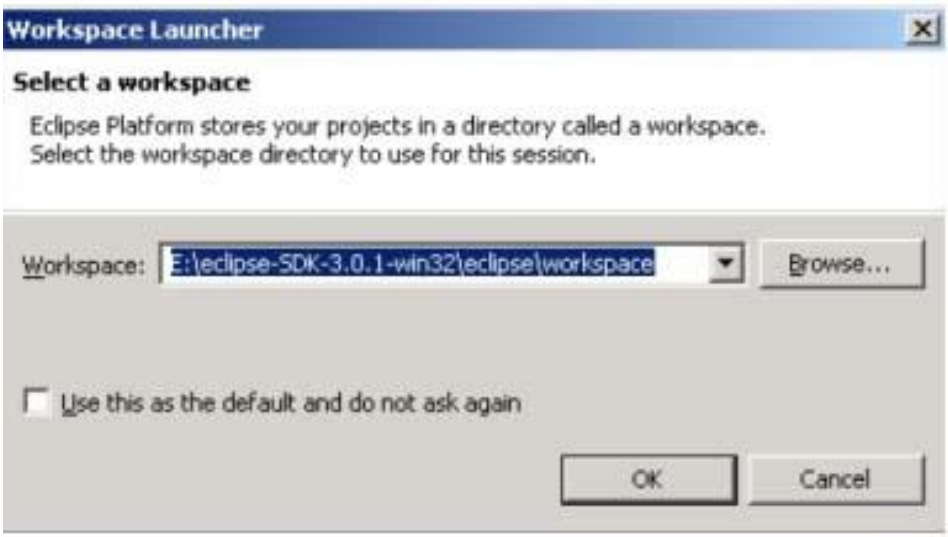

Figura 57. Directorio de trabajo.

Una vez seleccionada la ruta accederemos a la ventana principal de Eclipse y se mostrará una pantalla similar a la Figura 58.

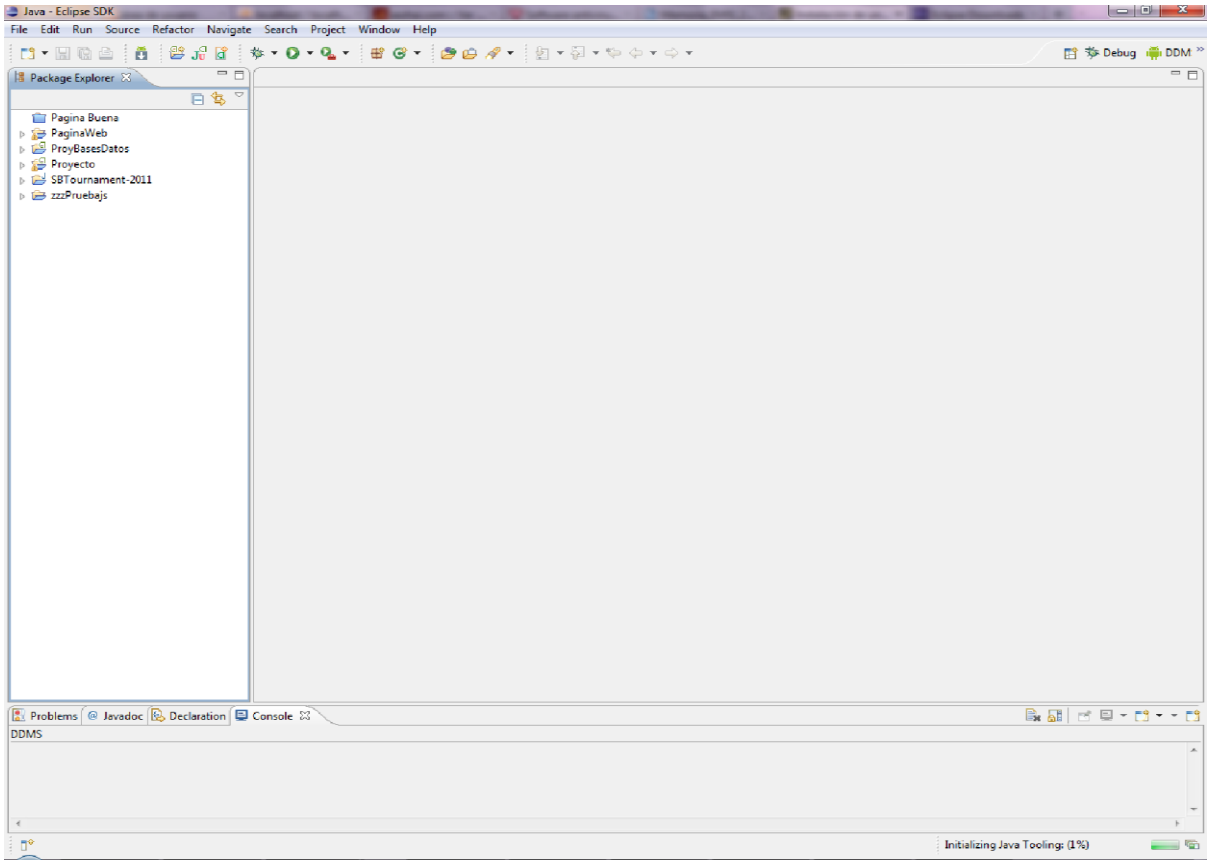

Figura 58. Ventana principal de Eclipse.

Una vez en esta ventana, que muestra la Figura 58, se podrá crear o abrir un proyecto y comenzar a trabajar con él. Con estos pasos ya hemos instalado el entorno de desarrollo Eclipse y estamos en total disposición de empezar a utilizarlo.

Apéndice 2. Instalación SDK

Para instalar el SDK de Android hay que descargar la última versión del SDK de Android accediendo a la siguiente página web:

http://code.google.com/android/download.html.

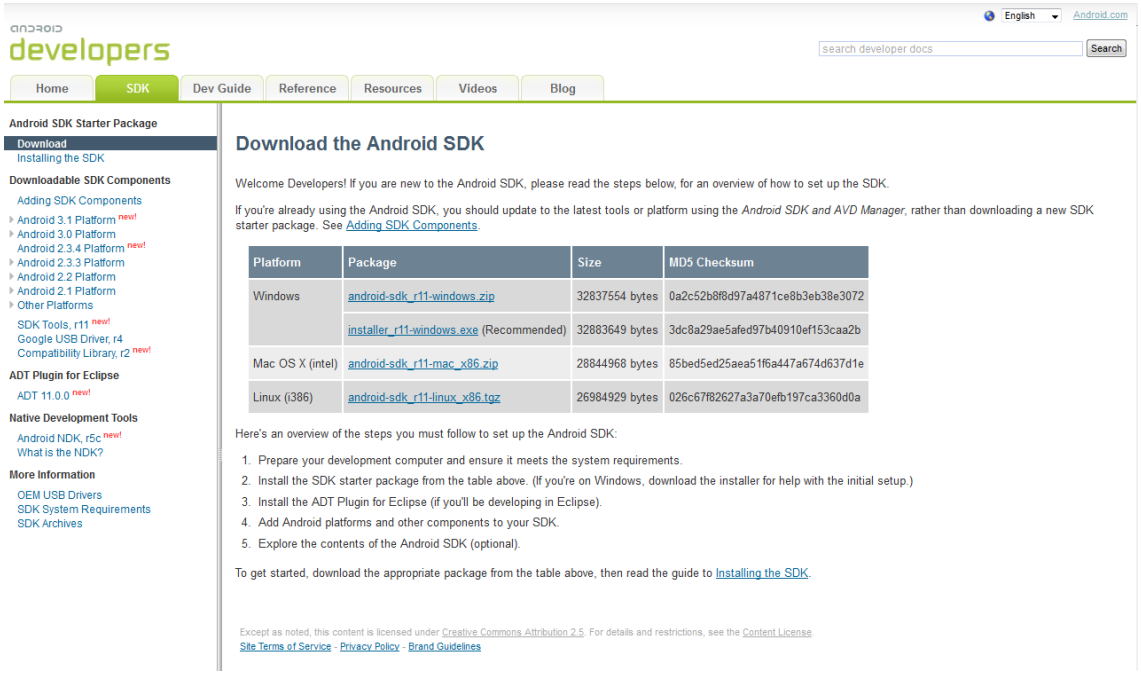

Figura 59. Página web del SDK de Android.

Puede elegir la última versión dependiendo de su sistema operativo Windows, Linux o Mac. Para el caso de Windows se recomienda la versión ejecutable para una instalación más sencilla. El archivo descargado es un archivo comprimido. Deberá descomprimir todos sus archivos en una nueva carpeta, cuya ruta debe recordarse para seguir unos pasos de configuración más adelante.

Una vez descomprimido los datos del archivo, el siguiente paso es acceder a la carpeta y instalar el SDK mediante el archivo SDK Manager.exe. El proceso de instalación tardará más de media hora dependiendo de su equipo.

Apéndice 3. Instalación del complemento ADT de Eclipse

Para instalar el Android Development Tools en la pantalla principal de Eclipse se debe seleccionar dentro del menú *Help* la opción *Install new software*. Aparece una nueva ventana de instalación que posibilita instalar nuevos plugins y software para Eclipse, debe pulsar en el botón *Add* del campo *Work with*. A continuación introduzca un nombre para el sitio y la siguiente *url:*

https://dl-ssl.google.com/android/eclipse

y pulse *OK*.

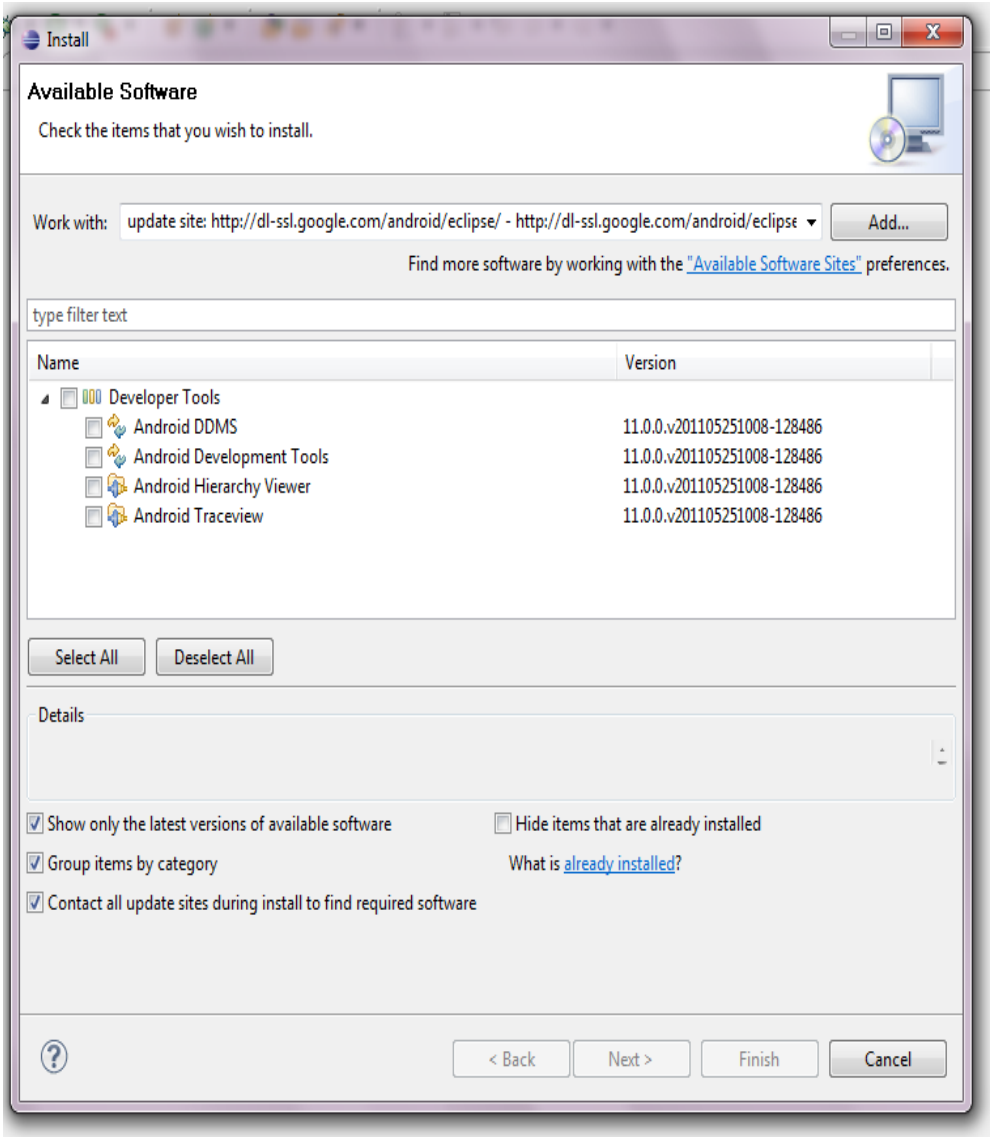

Figura 60. Instalación del ADT.

El siguiente paso es seleccionar todas las opciones que aparezcan disponibles y pulsar *Next*. Debe aceptar el contrato de licencia y pulsar *Next* para instalar el *plugin*. Por último comprobar y aceptar la ubicación de la instalación y pulsar *Finish*. Debe reiniciar Eclipse para el correcto funcionamiento.

Apéndice 4. Configuración del complemento Eclipse

Para configurar el complemento Eclipse seleccione la opción *Preferences* del menú *Window* de la pantalla principal de Eclipse.

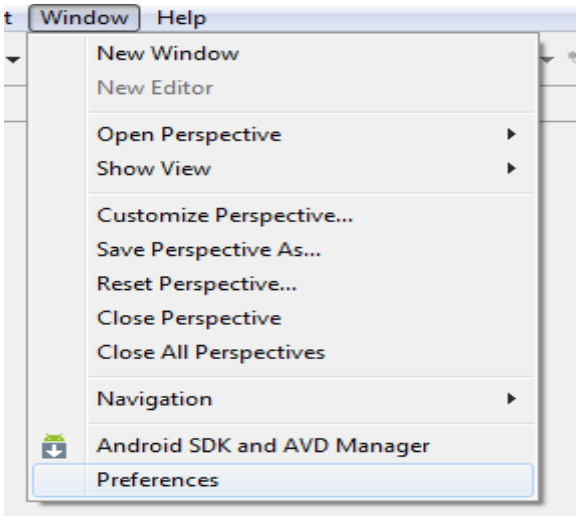

Figura 61. Menú Window.

Aparece la ventana que muestra la Figura 62, el siguiente paso es seleccionar la opción *Android* de la lista que aparece en el menú izquierdo.

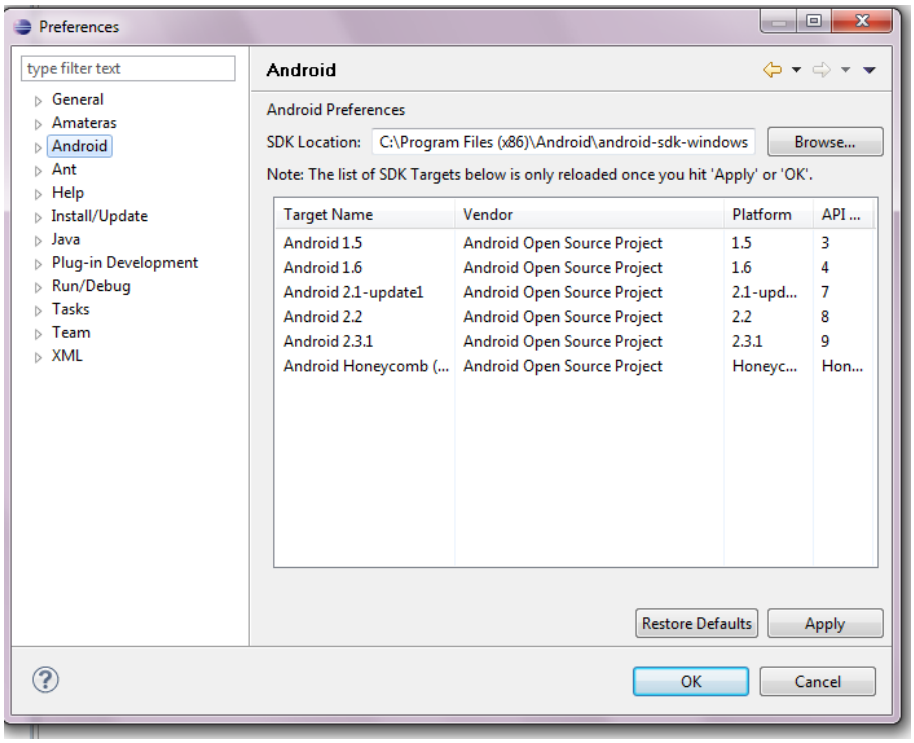

Figura 62. Configuración de las preferencias de Eclipse.

A continuación en la parte de la derecha de la ventana en el cuadro de opción *SDK Location* debe introducir la ubicación de la carpeta creada en el Anexo 2 tal y como se comentó que debía ser recordada. Al pulsar el botón *Apply* aparecen unas opciones, que hacen referencia a la versión de Android para la cual se quiere desarrollar la aplicación, bajo el nombre de *Target Name*, seleccione una de las versiones. Por último pulsar el botón *OK*, lo que nos llevara a Eclipse con el entorno de Android abierto y listo para empezar a programar.

Apéndice 5. Encuesta

Pregunta 1.

¿Hace usted la lista de la compra antes de acudir al supermercado?

Siempre Frecuentemente Alguna vez Nunca

Pregunta 2.

¿Usaría su móvil para llevar apuntada la lista de compra de una forma sencilla?

Prefiero hacer la lista en papel Prefiero hacer la lista en el móvil

Pregunta 3.

¿Le gustaría conocer de antemano el coste total de su lista una vez confeccionada?

Si me gustaría Me da igual Prefiero no saberlo

Pregunta 4.

¿Qué tipo de teléfono móvil utiliza?

Symbian iPhone Blackberry Android Otros

Pregunta 5.

¿Le parece interesante usar una aplicación móvil para facilitar su compra en el supermercado?

Muy interesante Indiferente Poco interesante

Pregunta 6.

Valore de 0 a 5 la aplicación móvil que le mostramos.

0 1 2 3 4 5

Capítulo 8
Bibliografía y referencias

1. Bibliografía

- Taha, Hamdy A., Investigación de Operaciones, Prentice Hall, 2004
- Narciso Martí, Yolanda Ortega, J. Alberto Verdejo, Estructura de datos y métodos algorítmicos, Prentice Hall, 2003.
- Frank Ableson, Charlie Collins, Robi Sen, Android, Guía para desarrolladores, Anaya, 2010
- La vanguardia, edición digital. Noticia publicada el día 1 de Junio 2011. Último acceso Junio 2011.

  http://www.lavanguardia.com/tecnologia/20110601/54164257864/android-lidera-el-mercado-de-las-aplicaciones-gratuitas.html
- Wikipedia, último acceso en Junio 2011.

  http://es.wikipedia.org/wiki/Wikipedia:Portada
- Campos Aucejo,Vicente() Problemas de Rutas, Dpto. Estadística i Investigació Operativa Universitat Valencia.

  http://www-eio.upc.es/~elena/Tutoriales/rutas.pdf
- Ciclo de vida de una aplicación, último acceso en Marzo 2011.

  http://celutron.blogspot.com/2007/12/ciclo-de-vida-de-una-aplicacin-android.html
- Descripción del ciclo de vida de una aplicación Android, por Google. Último acceso en Junio 2011.

  http://es.youtube.com/watch?v=fL6gSd4ugSI
- Guía para desarrolladores, Android Developers. Último acceso en Mayo 2011.

  http://developer.android.com/guide/index.html

- Android Developers, comunidad en Google Groups de desarrolladores para Androd (en inglés). Último acceso en mayo de 2011.

  http://groups.google.com/group/android-developers

- Desarrolladores-android, comunidad en Google Groups de desarrolladores sobre Android en español. Último acceso en Mayo 2011.

  http://groups.google.com/group/desarrolladores-android

- Android-Spa, principal comunidad de desarrolladores de Android en español. Último acceso en Mayo 2011.

  http://www.android-spa.com/

- SDK de Android. Web del proyecto Android. Último acceso en Noviembre 2010.

  http://code.google.com/android/download.html

- Descargas de distintos paquetes de Eclipse. Web oficial de la plataforma Eclipse. Último acceso en Noviembre 2010.

  http://www.eclipse.org/downloads/

- Guía de instalación del SDK de Android. Web del proyecto Android. Último acceso en Noviembre 2010.

  http://code.google.com/android/intro/installing.html

2. Referencias

[1] Android Market de Google:

https://market.android.com/

[2] Algoritmo de Dijkstra:

http://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra

[3] App Place de Toshiba:

http://apps.toshiba.com/

[4] Apple Computer:

http://www.apple.com/

[5] Casio:

http://www.casio-europe.com/es/

[6] Eclipse:

http://www.eclipse.org/

[7] El Corte Inglés: El Corte Inglés, plano de supermercado proporcionado por El Corte Inglés.

http://elcorteingles.es

[8] Hungry! , Pablo López-Jamar, Hungry! , Android Market

[9] Hewlett-Packard:

http://www.hp.com

[10] Itunes Store de Apple:

http://www.apple.com/es/itunes/

[11] Layar:

www.layar.com

[12] Microsoft:

www.microsoft.com

[13] Nintendo DS:

http://www.nintendo.es

[14] Open Handset Alliance:

http://www.openhandsetalliance.com/index.html

[15] Ovi Store de Nokia:

www.store.ovi.com

[16] Programación orientada a objetos con Java: David J. Barnes, Michael Kolling ; *Programación orientada a objetos con Java ;* Pearson Educación. 2007 3a edición;

[17] Research in Motion (RIM):

http://www.rim.com

[18] Shop Savvy Barcode Scanner:

http://shopsavvy.mobi/

[19] Snake:
http://developer.android.com/resources/samples/Snake/index.html

[20] Sony PlayStation Portable:

http://es.playstation.com/psp/

[21] Sun Microsystems:

http://www.oracle.com/us/sun/index.htm

[22] Versiones Android:
http://developer.android.com/resources/dashboard/platform-versions.html

[23] WebKit:

http: //www.webkit.org

[24] Wiktionary:
http://developer.android.com/resources/samples/Wiktionary/index.html